(19)日本国特許庁（ＪＰ）　(12)**公表特許公報（Ａ）**　(11)特許出願公表番号

特表平11－507752

(43)公表日　平成11年(1999)７月６日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	15/00	３３０	Ｇ０６Ｆ	15/00	３３０Ａ
	13/00	３５１		13/00	３５１Ｚ
	17/30		Ｇ０９Ｃ	1/00	６６０Ｅ
Ｇ０９Ｃ	1/00	６６０	Ｈ０４Ｌ	11/20	１０１Ｂ
Ｈ０４Ｌ	9/32			13/00	３０７Ｚ

審査請求　未請求　予備審査請求　有　(全 70 頁)　最終頁に続く

(21)出願番号　特願平9－503084
(86)(22)出願日　平成８年(1996)６月３日
(85)翻訳文提出日　平成９年(1997)12月５日
(86)国際出願番号　ＰＣＴ／ＵＳ９６／０７８３８
(87)国際公開番号　ＷＯ９６／４２０４１
(87)国際公開日　平成８年(1996)12月27日
(31)優先権主張番号　０８／４７４，０９６
(32)優先日　1995年６月７日
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０８／４８６，７９７
(32)優先日　1995年６月７日
(33)優先権主張国　米国（ＵＳ）

(71)出願人　オープン・マーケット・インコーポレーテッド
アメリカ合衆国，マサチューセッツ州　01803，バーリングトン，ウェイサイド　ロード　1
(72)発明者　レバーグッド・トーマス・マーク
アメリカ合衆国，マサチューセッツ州　01748，ホプキントン，ノース　ストリート　9
(74)代理人　弁理士　杉本　修司　(外１名)

最終頁に続く

(54)【発明の名称】　インターネットサーバーのアクセス管理およびモニタシステム

(57)【要約】
本発明はネットワークサーバーに対するアクセスを管理およびモニタする方法に関する。特に、本発明で説明された方法は、ハイパーテキストファイルを含むインターネットにおけるクライアント対サーバーセッションを含む。ハイパーテキスト環境では、クライアントは、ブラウザとして知られる標準プログラムによって、コンテンツサーバーにより伝送されたドキュメントを観察する。各ハイパーテキストドキュメントまたはページは、ユーザが横断を選択できる他のハイパーテキストページに対するリンクを含む。ユーザがアクセス管理ファイルに導かれるリンクを選択すると、サーバーはクライアントが認定または有効利用資格を有するか否かを決定する第２のサーバーにクライアントの要求を委ねる。このような確認があり次第、ユーザはセッション識別子を与えられ、それがユーザに現在の保護ドメイン内にある任意の他のファイルと同様に、要求ファイルにアクセスすることを可能にする。

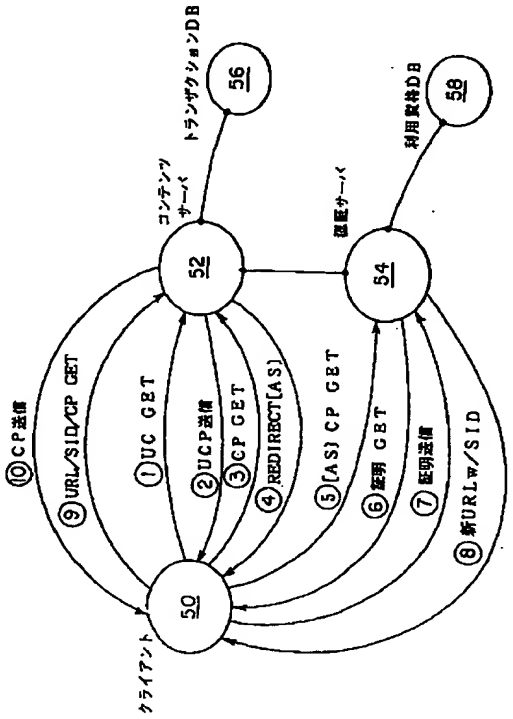

FIG. 3

【特許請求の範囲】

１．　ネットワークを介したクライアントからサーバーシステムへのサービス要求を処理する方法であって、

サービス要求を前記クライアントから前記サーバーシステムへ送出する段階と、

前記サーバーシステムから前記クライアントへセッション識別子を返送する段階と、

前記セッション識別子を前記要求およびその後に続く要求セッション内での前記クライアントから前記サーバーシステムへのサービス要求に付加する段階とを備えたサービス要求処理方法。

２．　請求項１において、前記サーバーシステムが前記要求セッション内にあるサービス要求のシーケンスのアクセス履歴を追跡するサービス要求処理方法。

３．　請求項２において、前記サーバーシステムが前記要求セッション内で行われた購入につながるサービス要求を決定するためのアクセス履歴を追跡するサービス要求処理方法。

４．　請求項１において、前記サーバーシステムが共通クライアントからの反復要求を除く特定のサービスに対する要求を計数するサービス要求処理方法。

５．　請求項１において、前記サーバーシステムがアクセスパターンに対する顧客情報であって、顧客の統計的データを含む情報に関するデータベースを維持するサービス要求処理方法。

６．　請求項１において、前記セッション識別子を発行するに先立ち、前記サーバーシステムが前記クライアントを認定ルーチンに供し、前記セッション識別子は偽造から保護されているサービス要求処理方法。

７．　請求項６において、前記サーバーシステムがサービス要求に対するセッション識別子を多数のサーバーに提供する認証サーバーを含む複数のサーバーを備えたサービス要求処理方法。

８．　請求項７において、

クライアントがサービス要求を要求されたサービスを行うべき最初のサ

ーバーに送信し、

前記最初のサーバーがセッション識別子に対するサービス要求を点検し、有効なセッション識別子を有するサービス要求のみにサービスを行い、サービス要求が無効な識別子を有する場合は、

前記最初のサーバーはクライアントからの前記サービス要求を前記認定サーバーに転送し、

前記認証サーバーはクライアントを認定ルーチンに供して前記サービス要求に付加される前記セッション識別子を最初のサーバーに発行し、

前記クライアントは前記セッション識別子が付加された前記サービス要求を前記最初のサーバーに送出し、

前記最初のサーバーは前記セッション識別子を認識し、前記クライアントに対して前記サービス要求のサービスを行い、

前記クライアントが前記サーバーシステムへの後続のサービス要求に対して前記セッション識別子を付加し、それにより追加の認定なしにサービスを受けるサービス要求処理方法。

９．　請求項１または７において、前記セッション識別子がユーザ識別子を含むサービス要求処理方法。

１０．　請求項１または７において、前記セッション識別子がセッションに対する満了時刻を含むサービス要求処理方法。

１１．　請求項７において、前記セッション識別子が、前記セッションがアクセス認定を有する保護ドメインに対するアクセスを提供するサービス要求処理方法。

１２．　請求項１１において、前記セッション識別子が異なる保護ドメインに対するアクセスのため修正されるサービス要求処理方法。

１３．　請求項７において、前記セッション識別子がキー管理に対する

キー識別子を提供するサービス要求処理方法。

１４．　請求項１または７において、前記サーバーシステムが前記セッション識別子からの情報を前記サーバーシステムにおけるトランザクションログに記録す

るサービス要求処理方法。

１５．　請求項１または７において、前記クライアントと前記サーバーシステム間の通信がハイパーテキスト転送プロトコル（ＨＴＴＰ）に従い、前記セッション識別子がＵＲＬ（ユニフォームリソースローケータ）におけるパス名の一部として付加されるサービス要求処理方法。

１６．　請求項１５において、前記クライアントが現在のＵＲＬのパス名を相対アドレス指定を用いて修正し、このパス名の前記セッション識別子部分をセッションにおける後続の要求に対して未修正のまま保持するサービス要求処理方法。

１７．　請求項１または７において、さらに、定まった時間間隔内にある特定のクライアントからの情報に対して作られた要求を除外する段階を備えたサービス要求処理方法。

１８．　ネットワークを介したクライアントからサーバーシステムへのサービス要求を処理する方法であって、

前記クライアントからネットワークを介して受信されたドキュメントに対する要求に応答する段階と、

ユーザ識別を含むセッション識別子を前記要求に付加する段階と、

前記セッション識別子のユーザ識別に基づき特定のユーザに対して個別化されている要求されたドキュメントを返送する段階とを備えたサービス要求処理方法。

１９．　ユーザにより購入されているクライアントからネットワークを介して受信されたドキュメントに対するサービス要求を処理する方法であって、

前記ユーザにより購入されているクライアントからネットワ

ークを介して受信されたドキュメントに対する要求に応答する段階と、

前記要求に対して認定識別子を付加する段階と、

前記ユーザが前記ドキュメントに対するアクセスを認定されていることを前記認定識別子が示す場合、要求された前記ドキュメントを返送する段階とを備えたサービス要求処理方法。

２０．　請求項１９において、前記要求に付加されているセッション識別子内に

前記認定識別子がコード化されるサービス要求処理方法。

２１．　ネットワークを介したクライアントからサーバーシステムへのサービス要求を処理する方法であって、

クライアントからネットワークを介して受信されたドキュメントに対する要求に応答する段階と、

ユーザ識別子を前記要求に付加する段階と、

要求された前記ドキュメントを前記クライアントに返送する段階と、

前記識別子で識別されたユーザに前記ドキュメントに対するアクセスに関する料金を請求する段階とを備えたサービス要求処理方法。

２２．　請求項２１において、前記要求に付加されているセッション識別子内でユーザ識別子がコード化されるサービス要求処理方法。

２３．　ネットワークを介したクライアントからサーバーシステムへのサービス要求を処理する方法であって、

サービス要求を前記クライアントから前記サーバーシステムへ送出する段階と、

セッション識別子を前記要求およびその後に続くセッション内にあるクライアントから要求のサーバーシステムへのサービス要求に付加する段階とを備えたサービス要求処理方法。

２４．　ネットワーク上の情報システムであって、

クライアントからのサービス要求を受信し、そのサービス要

求がセッション識別子を含んでいるか否かを決定する手段と、

要求セッションにおける初期サービス要求に対する応答にセッション識別子を提供する手段と、

前記セッション識別子を有するクライアントからのサービス要求およびそのセッションで処理される後続のサービス要求をサービスする手段とを備えた情報システム。

２５．　請求項２４において、前記セッション識別子を提供する手段が前記要求のサービスを行うサーバーシステムにある情報システム。

２６．　請求項２３において、さらに、前記セッション識別子を発行するに先立ち前記クライアントを認定する認定ルーチンと、前記セッション識別子を偽造から保護する手段とを備えた情報システム。

２７．　請求項２４において、さらに、前記セッション識別子からの情報を記録するトランザクションログを備えた情報システム。

２８．　請求項２４において、さらに、セッション内にあるサービス要求のシーケンスのアクセス履歴を追跡する手段を備えた情報システム。

２９．　請求項２４において、さらに、共通クライアントからの反復要求を除いて特定サービスに対する要求を計数する手段を備えた情報システム。

３０．　請求項２４において、アクセスパターンに対する顧客情報であって、顧客の統計的データを含む情報に関するデータベースを備えた情報システム。

３１．　請求項２５において、前記クライアントと前記サーバーシステム間の通信がハイパーテキスト転送プロトコル（ＨＴＴＰ）に従い、前記セッション識別子がＵＲＬにおけるパス名の一部として付加される情報システム。

３２．　ネットワーク上の情報サーバーであって、

ネットワークを介してクライアントから受信されたハイパーテキストページに対する要求にクライアントへ要求されたハイパーテキ

ストページを返送することで応答する手段と、

前記ハイパーテキストページにおけるリンクから導き出された追加要求に対して応答する手段と、

特定のハイパーテキストページから導き出された追加要求を追跡する手段とを備えた情報システム。

３３．　請求項３２において、前記要求が共通セッション識別子を含み、前記サーバーが要求セッション内にある要求を追跡する情報システム。

３４．　請求項３２において、さらに、アクセスパターンに対する顧客の統計的データに関するデータベースを備えた情報システム。

３５．　ネットワークを介したクライアントからサーバーシステムへの情報ページに対するアクセスを提供する方法であって、

前記クライアントの電話番号を提供する段階と、

電話番号を翻訳データベースを用いて目標ページ識別子にマッピングする段階と、

前記ページ識別子により記述された情報を前記サーバーシステムから要求する段階と、

前記ページ識別子により識別されたページを前記クライアント側で表示する段階とを備えた情報ページに対するアクセス方法。

３６．　ネットワークを介したクライアントからサーバーシステムへの情報ページに対するアクセスを提供する方法であって、

前記クライアント側で記述子を提供する段階と、

前記記述子を翻訳データベースを用いて目標ページ識別子にマッピングする段階と、

ユーザの追加行為なしに前記サーバーシステムからの前記ページ識別子により記述された情報をクライアント側で要求する段階と、

前記ページ識別子により識別されたページを前記クライアント側で表示する段階とを備えた情報ページに対するアクセス方法。

３７．　請求項３５または３６において、ＲＥＤＩＲＥＣＴコマンド中のＵＲＬをクライアントに返送してクライアントにＵＲＬを用いて情報を要求させるサーバーシステムにトランザクションデータベースが常駐する情報ページに対するアクセス方法。

３８．　請求項３６において、前記記述子が電話番号を備えた情報ページに対するアクセス方法。

３９．　請求項３６において、前記記述子が記述項目を備えた情報ページに対するアクセス方法。

４０．　請求項３９において、前記項目が会社名を含む情報ページに対するアクセス方法。

４１．　請求項３９において、前記項目が製品名を含む情報ページに対するアクセス方法。

４２．　請求項３９において、前記項目が音声マッピングにより識別される情報ページに対するアクセス方法。

４３．　請求項３５または３８において、前記目標ページ識別子が管理ページを記述している情報ページに対するアクセス方法。

４４．　請求項３５または３６において、前記目標ページ識別子がＵＲＬである情報ページに対するアクセス方法。

【発明の詳細な説明】

インターネットサーバーのアクセス管理およびモニタシステム

付属書類に対する言及

この特許文書の開示の一部は著作権保護の対象である末尾に添付の資料を含んでいる。この著作権者は特許開示内容の何者によるファクシミリ複製に対しても、それが特許商標局の特許ファイルまたは記録に現れるため異議を唱えるものではないが、さもない場合はいかなる形でもすべての著作権を留保する。

発明の背景

１９６０年代後半に開始されたインターネットは、地球全体に拡がる多くの小規模ネットワークから成る広大なコンピュータネットワークである。インターネットは指数関数的に成長しており、個人から法人の範囲にわたる数百万のユーザが今や専用線およびダイヤルアップ接続を利用して世界中で日常的にインターネットを使用している。“ホスト”として知られるインターネット内で接続されたコンピュータまたはそのネットワークが、殆どあらゆる専門分野における情報を持つデータベースへの一般のアクセスを可能にしており、大学および政府から多くの民間組織にまでわたる事業により支持されている。

インターネット上の情報は“サーバー”を介して一般が利用可能になる。サーバーはインターネットホスト上でそのホスト内に含まれるファイルまたはドキュメントを利用可能にするために作動しているシステムである。このようなファイルは代表的には、ローカルからホストにかけて、テープドライブまたは固定ディスクのような磁気記憶装置上で記憶される。インターネットサーバーは、ホスト上のファイルを要求するあらゆるコンピュータに情報を配布することができる。このような要求をするコンピュータは“クライアント”として知られ、これはインターネットに接続されたワークステーション、掲示板システムまたは家庭用パーソナルコンピュータ（ＰＣ）であり得る。ＴＣＰ／ＩＰ（伝送制御プロトコル／インターネットプロトコル）は、インターネットの完全な使用を可能にするネットワークプロトコルの一種である。ＴＣＰ／ＩＰネットワーク上のすべてのコンピュータが独自のＩＤコードを必要とする。従って、インターネット上の各コ

ンピュータまたはホストは、ＩＰ（インターネットプロトコル）番号またはアドレスとして知られた、ネットワークおよびコンピュータの名称に対応する独自の番号コードにより識別される。過去には、インターネットユーザは要求するファイルを指定するため、ホストコンピュータおよびホストの記憶装置内のディレクトリのパスを識別することによってのみ、そのリソースにアクセスすることができた。様々なナビゲーティングツールが、特定のホストアドレスを知ることなしにイターネット上のリソースを検索するためユーザを支援しているものの、これらのツールはインターネットに関する多分に技術的な知識を要する。

ワールドワイドウェブ（ウェブ）は、ＩＰアドレスまたは他の技術的知識なしで直観的にユーザがインターネットリソースをナビゲートすることを可能にするインターネット上の情報をアクセスする一方法である。ウェブは、代表的にはインターネットサーバーと通信するためのコマンドのセットを伝送することをユーザに要求するコマンドラインユーティリティなしで利用できる。代わりに、ウェブはコンピュータモニタに表示可能な数百数千の相互接続された“ページ”、またはドキュメントから構成される。ウェブページは特別なサーバーを動かしているホストにより供給される。これらのウェブサーバーを作動するソフトウェアは比較的簡単で、ＰＣを含めた広範囲のコンピュータプラットフォーム上で利用可能である。同程度に利用可能なものはウェブ“ブラウザ(browser)”として知られ、クライアントシステム上の従来の非ウェブファイルと同様にウェブページを表示するために用いられるクライアントソフトウェアの一形式である。今日、ウェブサーバーを提供するインターネ

ットホストは、月に３００を超える割合で増加しており、インターネット通信の好ましい方法となりつつある。

１９９１年に創設されたウェブは、“ハイパーテキスト”および“ＨＴＴＰ”（ハイパーテキスト転送プロトコル）として知られる転送方式の概念に基づいている。ＨＴＴＰは主としてＴＣＰ／ＩＰ上で作動するよう設計され、標準的なインターネットセットアップを用い、そこでサーバーがデータを供給し、クライアントがそれを表示または処理する。情報転送のためのフォーマットの１つは、ハ

イパーテキスト記述言語（ＨＴＭＬ）を用いてドキュメントを作成することである。ＨＴＭＬページは、ページがどのように表示されるべきかを指示するフォーマットコードのほかに、標準的なテキストから構成される。ウェブクライアント、ブラウザはページを表示するためこれらのコードを読み取る。ハイパーテキスト協定およびワールドワイドウェブの関連機能は、本明細書で参考のため引用された、Ｐａｙｎｅらにより１９９４年１０月２４日に出願された米国特許出願第０８／３２８，１３３号の付属書類に記述されている。

各ウェブページはテキストに加えて画像および音声を含むことがある。特定のテキストの背後に隠れて、画像または音声は、“ハイパーテキストリンク”（“リンク”）として知られる、同一のサーバー内、またはインターネット内の他のコンピュータ上の他のページに対して接続されたものである。例えば、リンクは、下線が引かれるか第２の色彩で表示されることがある単語または文節として視覚的に表示することができる。各リンクはＵＲＬ（Uniform Resource Locator）と呼ばれる特別な名前を用いて、あるウェブページに導かれている。ＵＲＬは、ウェブブラウザが任意のウェブサーバー上に保持された任意のファイルに直接アクセスすることを可能にする。ユーザはまた、他のウェブページにジャンプするため、既知のＵＲＬをウェブページ上のコマンドラインに直接書き込むことにより指定することができる。

ＵＲＬネーミングシステムは３つの部分、すなわち、転送フォーマット、ファイルを保持する機械のホスト名、ファイルへのパスからなる。ＵＲＬの一例は以下の通りである。

ｈｔｔｐ：／／ｗｗｗ．ｃｏｌｌｅｇｅ．ｕｎｉｖ．ｅｄｕ／Ａｄｉｒ／Ｂｄｉｒ／Ｃｄｉｒ／ｐａｇｅ．ｈｔｍｌ

ここに“ｈｔｔｐ”は、転送プロトコルを表す。コロンおよび２本のスラッシュ（：／／）は、転送プロトコルをホスト名から離すために用いられる。“ｗｗｗ”は要求されているファイルがウェブページであることを意味し、“ｗｗｗ．ｃｏｌｌｅｇｅ．ｕｎｉｖ．ｅｄｕ”はホスト名である。“／Ａｄｉｒ／Ｂｄｉｒ／Ｃｄｉｒ”はホストマシン上のツリー構造、またはパスにおけるディレクト

リ名の一セットである。“ｐａｇｅ．ｈｔｍｌ”はファイルがＨＴＭＬで書き込まれていることを示すファイル名である。

インターネットは、情報の交換が制限なしに無料で行われるオープンストラクチャーを維持する。しかしながら、インターネットに特有のフリーアクセス形式は、自己のインターネットサーバーの管理が必要である情報提供者に困難を呈している。例えば、特定の技術情報を自己のインターネットサーバー上で世界中の大規模なグループの仲間たちに利用可能とすることを希望するが、その情報の機密性が保たれなければならない研究組織を考えてみよう。各クライアントを識別する手段なしでは、その組織はネットワーク上で情報を機密的または選択的な形で与えることができないであろう。他の状況では、会社が極めて特殊なサービスの情報を自己のインターネットサーバー上でサービス契約または利用資格を有する顧客に対してのみ提供することを希望する場合がある。

インターネットサーバーによるアクセス管理は少なくとも２つの理由で困難である。第１に、クライアントが遠隔のインターネットサーバー上のファイルに要求を送信すると、このメッセージは、宛て先ホストに到達するまで、インターネットを介して接続されたコンピュータのウェ

ブにより通信経路選択または中継される。クライアントはそのメッセージがどのようにサーバーに到達するかを知る必要はない。同時に、サーバーは、クライアントが誰で、ＩＰアドレスが何であるかを正確に知ることさえなく応答を行う。サーバーがそのクライアントを追跡するようにプログラミングされ得るものの、追跡のタスクはそれが不可能とまではいかないが、しばしば困難である。第２に、構内用のローカルエリアネットワーク（ＬＡＮ）に対する望ましくない侵入を防ぐため、システムアドミニストレータはそのネットワーク内に、インターネット“ファイアウォール”のような様々なデータフロー制御メカニズムを実装する。インターネットファイアウォールは、ユーザがインターネットに匿名で到達することを可能にする一方、外部の侵入者がユーザのＬＡＮにアクセスすることを防止する。

発明の内容

本発明は、ネットワークを介したクライアントからサーバーへのサービス要求を処理する方法に関する。とりわけ本発明は、ワールドワイドウェブ（ウェブ）のようなＨＴＴＰ（ハイパーテキスト転送プロトコル）環境におけるクライアント要求を処理することに適する。本発明の一構成は、サービス要求をクライアントからサーバーに送出し、セッション識別子（ＳＩＤ）をその要求、およびその後に続く要求セッション内でのクライアントからサーバーに対するサービス要求に対して付加することを含む。好ましい実施形態では、本発明の方法はクライアントにより作られた最初のサービス要求におけるＳＩＤをサーバーからクライアントに返送することを含む。有効なＳＩＤは、ユーザが管理されたファイルにアクセスすることを可能にする認定(authorization)識別子を含むことがある。

好ましい実施形態では、クライアント要求はUniform Resource Locator（ＵＲＬ）によりウェブブラウザから作られる。クライアント

要求が管理されたファイルにＳＩＤなしで送信されると、インターネットサーバーはクライアントをＳＩＤ発行前に認定ルーチンにかけ、ＳＩＤは偽造から保護される。コンテンツサーバーはクライアントの要求を異なるホストに存在し得る認証サーバーに転送することにより認定ルーチンを開始する。転送された要求を受信すると、認証サーバーはクライアントに質問するため応答を返し、次いで資格を付与されたクライアントに対してＳＩＤを発行する。新規のクライアントに対して、認証サーバーは新規の利用資格（アカウント）を開いてから後にＳＩＤを発行することがある。有効なＳＩＤは、代表的には、ユーザ識別子、アクセス可能なドメイン、キー識別子、日付のような満了時刻、ユーザコンピュータのＩＰアドレス、および秘密キーで暗号化されたＳＩＤにおける他の項目すべての暗号ハッシュのような記憶された電子署名から構成される。認証サーバーは、次いでこのＳＩＤが付加されたオリジナルＵＲＬから成る新しい要求を、ＲＥＤＩＲＥＣＴによってクライアントに送出する。新しいＵＲＬにより形成された修正要求は、自動的にクライアントブラウザによりコンテンツサーバーに送出される。

コンテンツサーバーがＳＩＤを伴うＵＲＬ要求を受信すると、コンテンツサーバーはＳＩＤを伴うＵＲＬおよびユーザのＩＰアドレスをトランザクションログ

(log)中に記録し、ＳＩＤを有効とする段階に進む。ＳＩＤがそのように有効になると、コンテンツサーバーは要求されたドキュメントをクライアントのウェブブラウザによる表示のために送信する。

好ましい実施形態では、有効ＳＩＤは、クライアントが追加の認定を要求することなく保護ドメイン内にあるすべての管理ファイルにアクセスすることを可能にする。保護ドメインは、サービスプロバイダにより定義され、１つまたは複数のサーバー内における共通保護の管理ファイルの集まりである。

クライアントが有効ＳＩＤで管理ウェブページをアクセスすると、そ

のページを見ているユーザが別のウェブページを見るためリンクを横断することを望む場合がある。この時、幾つかの可能性がある。ユーザはリンクを横断して同じパスにある別のページに達することができる。これは“相対リンク”と呼ばれる。相対リンクは同じドメイン内、或いは異なるドメインに対して作られることができる。クライアントコンピュータ上のブラウザは、古い管理ページを新しい管理ページと交換するため、現在のＵＲＬを書き直すことにより相対リンクを実行、つまりリンクの横断を行う。新しいＵＲＬは新しいページ名を除き、ＳＩＤを含む古いＵＲＬのすべての部分を保持する。相対リンクが同じ保護ドメインのページを示すものであれば、ＳＩＤは有効のままであり、要求は受理される。しかしながら、相対リンクが異なる保護ドメインのページを示すものであれば、ＳＩＤはもはや有効ではなく、クライアントは自動的にＳＩＤを更新するため書換ＵＲＬを認証サーバーに送出するよう転送される。更新された、または新しいＳＩＤは、ユーザの資格が認定されれば、新しいドメインに対するアクセスを与える。

ユーザは、リンクを横断して異なるパスにおけるドキュメントに達することも選択できる。これは“絶対リンク”と呼ばれる。絶対リンクを生成する際、ＳＩＤはブラウザにより上書きされる。好ましい実施形態では、コンテンツサーバーがドメイン内の管理ウェブページの各供与に際して、ページをフィルタリングして、現ＳＩＤをページ上の各絶対ＵＲＬに含ませる。従って、ユーザが絶対リンクを横断することを選択すると、ブラウザは異なるパスにおけるページにそのＳ

ＩＤと共に送信される認定されたＵＲＬにより支援される。別の実施形態では、コンテンツサーバーは上記のようなフィルタリング手順なしで作動し、更新のために絶対ＵＲＬを認証サーバーに転送することができる。

絶対リンクはまた、異なるドメインにおける管理ファイルに導かれることがある。さらに、このような要求は新規のＳＩＤを処理するために認証サーバーに転送される。管理されていないファイルに導かれる絶対

リンクは、即時アクセスを可能にする。

別の実施形態では、サーバーアクセス管理はクライアントブラウザをプログラミングして、この特定サーバーに対する各ＵＲＬコールにおける使用のためにＳＩＤまたは類似のタグを記憶することにより維持できる。しかしながら、この実施形態は、このような通信を取り扱うウェブに共通な標準ブラウザ形式に一般に適さない特殊なブラウザを要する。

本発明の他の構成は、管理された、または管理されていない双方の様々なページに対するアクセスの頻度およびアクセス継続時間をモニタすることである。コンテンツサーバー内のトランザクションログは、ページをアクセスしたリンクシーケンスを含む、ページに対する各クライアントのアクセス履歴を保持する。さらに、コンテンツサーバーはクライアント要求を、共通のクライアントから繰り返された要求を除いてカウントすることがある。このような記録は、ユーザの要望、アクセスパターン、および顧客の統計的データとアクセスされたページとアクセスパターンとの関係ような重要なマーケティングフィードバックを提供する。

構成の種々の新規な詳細および各部分の組み合わせを含む本発明の上記特徴およびその他の特徴は、添付図面を参照して説明され、請求の範囲に記載されている。本発明を具体化する特定の装置および方法は、例として示されており、本発明はこれに限定されないことが理解されよう。本発明の主要部および特徴は、本発明の範囲を逸脱することなく様々な数多くの実施形態で具体化することができる。

図面の簡単な説明

図１は、インターネット運用を示す線図である。

図２Ａは、インターネットサーバーアクセス管理およびモニタの好ましい方法を説明するフローチャートである。

図２Ｂは、認証プロセスの詳細を説明する関連フローチャートである

。

図３は、本発明のアクセス管理およびモニタ方法を含むクライアント対サーバーの取り交わしセッションの例を示す線図である。

図４は、ワールドワイドウェブページの例を示す図である。

図５は、認定書式ページを示す図である。

図６は、電話番号のＵＲＬに対する翻訳の詳細を示す線図である。

好ましい実施形態の詳細な説明

図を参照すると、図１はインターネットの線図である。インターネット１０は、コンピューサーブまたはアメリカオンラインのようなインターネットプロバイダ１６および情報システム（ＢＢＳ）２０によって所有されるシステムを含む数百万の相互接続されたコンピュータ１２のネットワークである。個人または法人ユーザは、幾つかの方法でインターネットに対する接続を確立することができる。家庭用ＰＣ１４上のユーザは、インターネットプロバイダ１６を介してアカウントを購入することができる。モデム２２を用い、ＰＣユーザはインターネットプロバイダにダイヤルして高速モデム２４に接続することができ、そのモデムが次いでインターネットに対する完全なサービス接続を行う。ユーザ１８は、その顧客に対してインターネットゲートウェイ接続を提供するＢＢＳ２０を介して、インターネットに対する幾分か制限された接続を行うことがかできる。

図２Ａは本発明の好ましい処理の詳細を示すフローチャートであり、図４はブラウザによりクライアント側で表示されるサンプルウェブページの図である。ページは下線付きリンクテキスト４１２を含むテキスト４０４を備える。タイトルバー４０８およびＵＲＬバー４０２はそれぞれ、現在のウェブページのタイトルおよびＵＲＬを表示する。図４に示されるように、ページのタイトルは“CONTENT HOME PAGE”であり、それに対応するＵＲＬは“ｈｔｔｐ：／／ｃｏｎｔｅｎ

ｔ．ｃｏｍ／ｈｏ

ｍｅｐａｇｅ”である。カーソル４１４がリンクテキスト４１２ｂの上に置かれた時、マウスをクリックすることで探し出されるページは、代表的にはそのリンクに対するＵＲＬを示すステータスバー４０６で識別される。この例では、指示リンク４１２ｂに対するＵＲＬが、“コンテンツ(content)”と呼ばれる商業コンテンツサーバーにおける“広告(advertisement)”と呼ばれるページに導かれることを、ステータスバー４０６は示す。リンクテキストをクリックすることで、ユーザはブラウザに図２Ａの１００におけるＵＲＬ　ＧＥＴ要求を生成させる。ブラウザはその要求をコンテンツサーバー１２０に送出し、コンテンツサーバー１２０は要求されたページが管理ドキュメントであるか否かを決定する最初の判断１０２によって処理する。その要求が未管理ページに導かれる場合、この例の“広告”ページにおけるように、コンテンツサーバーはＵＲＬおよびＩＰアドレスを入手できる範囲でトランザクションログ１１４に記録する。コンテンツサーバーは次いで、要求されたページをユーザコンピュータ上に表示するためブラウザに送信する（１１６）。

その要求が管理ページに導かれる場合、コンテンツサーバーはＵＲＬがＳＩＤを備えるか否かを決定する（１０４）。例えば、ＵＲＬは、ＳＩＤを必要とする“ｈｔｔｐ：／／ｃｏｎｔｅｎｔ．ｃｏｍ／ｒｅｐｏｒｔ”のような管理ページ名“リポート(report)”に導かれる。ＳＩＤが存在しない場合、コンテンツサーバーは“ＲＥＤＩＲＥＣＴ”応答１２２をブラウザ１００に送信し、ユーザの初期要求を認証サーバー２００に転送して有効ＳＩＤを入手する。認証処理の詳細は図２Ｂに説明されており、後に論議されるが、処理結果は認証サーバーからクライアントに提供されるＳＩＤとなる。上記の例では、ＳＩＤが付加された修正ＵＲＬは、“ｈｔｔｐ：／／ｃｏｎｔｅｎｔ．ｃｏｍ／〔ＳＩＤ〕／ｒｅｐｏｒｔ”となり得る。好ましいＳＩＤは、１文字当たり６ビットの９６ビットＳＩＤデータをコード化する１６文字ＡＳＣＩＩ文字列であ

る。これは３２ビット電子署名、１時間単位の１６ビット満了日付、キー管理に

用いられる２ビットキー識別子、現ＳＩＤがアクセスを認定する情報ファイルセットを備える８ビットドメイン、および２２ビットユーザ識別子を含む。残りのビットは拡張のため蓄えられる。電子署名は、認証サーバーとコンテンツサーバーにより共有されている秘密キーにより暗号化されているＳＩＤおよび認定ＩＰアドレスにおける残留項目を用いた暗号ハッシュである。

初期ＧＥＴ　ＵＲＬがＳＩＤを含む場合、コンテンツサーバーは、要求が現ドメイン内のページに導かれるか否かを決定する（１０６）。ＳＩＤを有する要求が異なるドメインの管理ページへ導かれる場合、ＳＩＤはもはや有効ではなく、ユーザは認証サーバーに再度導かれる（１２２）。

要求が現ドメイン内の管理ページに対するものである場合、コンテンツサーバーはＳＩＤを伴う要求ＵＲＬ、およびユーザＩＰアドレスをトランザクションログ１０８に記録するように手順を進める。コンテンツサーバーは次いでＳＩＤを有効化する（１１０）。このような有効化は次の点検リストを含む。（１）ＳＩＤの電子署名が、認証サーバーとコンテンツサーバーにより共有されている秘密キーを使用して、ＳＩＤおよび認定ＩＰアドレスにおける残留項目から計算された電子署名と比較される。（２）ＳＩＤのドメインフィールドが、認定されたドメイン内にあることを確認するため点検される。（３）ＳＩＤのＥＸＰフィールドが現在時刻よりも後であることを確認するため点検される。

有効性が認められると、コンテンツサーバーは、送出されるページをサーバーに含まれる絶対ＵＲＬリンク、すなわち、異なるコンテンツサーバーにおける管理ドキュメントに導かれる任意リンクについて探索する（１１２）。コンテンツサーバーは、各絶対ＵＲＬを現ＳＩＤとともに増加し、多数のコンテンツサーバーにわたる認証されたアクセスを容易にする。要求されたページは、処理されたとおりの形で、次いで表示

のためクライアントブラウザに伝送される。要求されたウェブページを見ているユーザは、そのページ上の任意リンクを横断して全シーケンスを再度誘発するように選択することができる（１００）。

図２Ｂは認証処理の詳細を説明する。コンテンツサーバーはクライアントを認

証サーバーへ再度導くことがある。ＲＥＤＩＲＥＣＴ　ＵＲＬは以下の形をとり得る。

“ｈｔｔｐ：／／ａｕｔｈ．ｃｏｍ／ａｕｔｈｅｎｔｉｃａｔｅ？ｄｏｍａｉｎ＝〔ｄｏｍａｉｎ〕&ＵＲＬ＝ｈｔｔｐ：／／ｃｏｔｅｎｔ．ｃｏｍ／ｒｅｐｏｒｔ”

ＵＲＬは認証を要求し、ドメインおよび初期ＵＲＬを指定する。ＲＥＤＩＲＥＣＴに対する応答で、クライアントブラウザは自動的にＧＥＴ要求を、用意されたＵＲＬと共に送信する。

コンテンツサーバーがクライアントを認証サーバー２００に再度導く時は何時でも、認証サーバーは、それが承認されたコンテンツサーバーに対するものであることを確認し、要求されたアクセスに対して必要な認証のレベルを決定することにより、認定処理を開始する（２１０）。このレベルに応じて、サーバーはユーザに証明を要求することができる（２１２）。要求が低レベルドキュメントに対するものである場合、認証サーバーは適切なＳＩＤを直ちに発行し（２２８）、証明点検手順に移る。ドキュメントが証明を要する場合、認証サーバーは、“ＣＨＡＬＬＥＮＧＥ”応答を送信して、クライアントブラウザがユーザに確認を促すよう仕向ける（２１４）。好ましい確認問合せは、代表的にはユーザ名およびパスワードの要求から成る。ユーザがパスワードを与えることができない場合、そのアクセスは拒否される。ブラウザは与えられた情報から認定ヘッダ３００を形成し、認定ヘッダと共に最終ＵＲＬを用いてＧＥＴ要求を認証サーバーに再度送信する。例えば、このようなＧＥＴ要求のＵＲＬは次の形をとり得る。

“ｈｔｔｐ：／／ａｕｔｈ．ｃｏｍ／ａｕｔｈｅｎｔｉｃａｔｅ？ｄｏｍａｉｎ＝〔ｄｏｍａｉｎ〕&ＵＲＬ＝ｈｔｔｐ：／／ｃｏｔｅｎｔ．ｃｏｍ／ｒｅｐｏｒｔ”および認定ヘッダは“ＡＵＴＨＯＲＩＺＥ：〔ａｕｔｈｏｒｉｚａｔｉｏｎ〕”であり得る。

ＧＥＴ要求を受信次第、要求ドキュメントにアクセスすることをユーザが認定されているか否かを決定するため（２１８）、認証サーバーは利用資格データベースに問い合わせる（２１６）。好ましい利用資格データベースは、ユーザの年

齢、住所、趣味、または職業のような、コンテンツサーバーが後に使用するユーザの統計的データ情報のほかに、クライアントのＩＰアドレスおよびパスワードのような識別目的のための情報を含むユーザのプロファイルを包含することがある。ユーザが認定されると、ＳＩＤが前に説明されたように生成される（２２８）。ユーザが認定されない場合、認証サーバーは、ユーザが新しい利用資格に対する資格を有するか否かを点検する。ユーザが新しい利用資格を開く資格を持たない場合、アクセスを拒否するページがクライアントブラウザ１００に伝送される（２２２）。ユーザが資格を有する場合、新しいユーザには図５に図示されたようなリアルタイムオンライン登録を開始する書式ページが伝送される（２２４）。この書式は、例えば、ユーザの個人情報およびクレジット照合を要する。ブラウザは認証サーバーに対する“ＰＯＳＴ”メッセージとして空白５０２にユーザにより記入されたデータを伝送することができる。ＰＯＳＴメッセージは書式のコンテンツをＵＲＬの一部としてではなく、データ部でサーバーに送信されるようにする。新しいユーザにより記入された登録書式が有効になると（２２６）、適切なＳＩＤが生成される（２２８）。登録が有効でない場合、アクセスは再度拒否される（２２２）。

認定ユーザに対するＳＩＤが、コンテンツサーバー上の管理ページに導かれたオリジナルＵＲＬに付加される（“タグ付け”）（２３０）。認証サーバーは次いでＲＥＤＩＲＥＣＴ応答をタグ付きＵＲＬに基づき、クライアントブラウザ１００に伝送する（２３２）。“ｈｔｔｐ：／

／ａｕｔｈ．ｃｏｍ／〔ＳＩＤ〕／ｒｅｐｏｒｔ”のような修正されたＵＲＬが、自動的にコンテンツサーバー１２０に送出される。

図３は、本発明のアクセス管理およびモニタ方法を含む代表的なクライアント対サーバーの取り交わしを図示する。段階１において、ブラウザを作動しているクライアント５０は、ネットワークを介して未管理ページ（ＵＣＰ）に対するＧＥＴ要求を伝送する。例えば、ユーザは“ｃｏｎｔｅｎｔ．ｃｏｍ”がサーバー名および“広告(advertisement)”が未管理ページ名である、ＵＲＬ“ｈｔｔｐ：ｃｏｎｔｅｎｔ．ｃｏｍ／ａｄｖｅｒｔｉｓｅｍｅｎｔ”を伝送することで、

広告ページを要求することができる。段階２では、コンテンツサーバー５２はＧＥＴ要求を処理して要求ページ、“広告(advertisement)”を伝送する。コンテンツサーバーはまた、ＵＲＬ、クライアントのＩＰアドレス、および現在時刻を記録することでＧＥＴ要求をトランザクションデータベース５６に記録する。

段階３では、クライアントマシン上のユーザが、管理ページ（ＣＰ）に導かれた広告ページ中のリンクを横断するように選択できる。例えば、広告ページは“リポート（report）”と呼ばれる管理ページに対するリンクを含むことがある。このリンクを選択すると、クライアントブラウザ５０はＧＥＴ要求をリポートファイル“ｈｔｔｐ：／／ｃｏｎｔｅｎｔ．ｃｏｍ／ｒｅｐｏｒｔ”が関連するＵＲＬを介して送出する。コンテンツサーバー５２は、要求が管理ページに対するものであること、およびＵＲＬがＳＩＤを含んでいないことを決定する。段階４では、コンテンツサーバーはＲＥＤＩＲＥＣＴ応答をクライアントに伝送し、さらに、段階５では、ブラウザは自動的にＲＥＤＩＲＥＣＴ　ＵＲＬを認証サーバー５４に送信する。認証サーバーに送信されたＲＥＤＩＲＥＣＴ　ＵＲＬは下記の文字列を含み得る。

“ｈｔｔｐ：／／ａｕｔｈ．ｃｏｍ／ａｕｔｈｅｎｔｉｃａｔｅ？ｄｏｍａｉｎ＝〔ｄｏｍａｉｎ〕＆ＵＲＬ＝ｈｔｔｐ：／／ｃｏｔｅｎｔ．

ｃｏｍ／ｒｅｐｏｒｔ”

認証サーバーはＲＥＤＩＲＥＣＴを処理して、認定のためにユーザの確認（ＣＲＥＤ）が必要か否かを決定する。段階６では、認証サーバーは“ＣＨＡＬＬＥＮＧＥ”応答をクライアントに伝送する。前述のように、代表的な確認はユーザ名とパスワードから成る。確認情報に基づく認定ヘッダは、次いでクライアントブラウザにより認証サーバーへ送信される。例えば、このような認定ヘッダを有するＧＥＴ　ＵＲＬは“ｈｔｔｐ：／／ａｕｔｈ．ｃｏｍ／ａｕｔｈｅｎｔｉｃａｔｅ？ｄｏｍａｉｎ＝〔ｄｏｍａｉｎ〕＆ＵＲＬ＝ｈｔｔｐ：／／ｃｏｔｅｎｔ．ｃｏｍ／ｒｅｐｏｒｔ”であり、認定ヘッダは“ＡＵＴＨＯＲＩＺＥ：〔ａｕｔｈｏｒｉｚａｔｉｏｎ〕”であり得る。認証サーバーはＧＥＴ要求を利用資格データベース５８を点検することで処理する。有効な利用資格がユーザに対し

て存在する場合、ＳＩＤが発行され、それが管理ページ“リポート（report）”およびドメイン内の他のすべてのページに対するアクセスを認定する。

前述のように、好ましいＳＩＤはユーザ識別子、現在のドメイン、キー識別子、満了日時、クライアントのＩＰアドレス、および記憶された電子署名をコード化するコンパクトなＡＳＣＩＩ文字列から構成される。段階８では、認証サーバーはクライアントをタグ付きＵＲＬ、“ｈｔｔｐ：／／ｃｏｔｅｎｔ．ｃｏｍ／〔ＳＩＤ〕／ｒｅｐｏｒｔ”に再度導く。段階９では、タグ付きＵＲＬは自動的にブラウザによりコンテンツサーバーにＧＥＴ要求として送信される。コンテンツサーバーはタグ付きＵＲＬ、クライアントのＩＰアドレス、および現在時刻を記録することでＧＥＴ要求をトランザクションデータベース５６に記録する。段階１０では、コンテンツサーバーが、ＳＩＤの有効性を確認次第、要求された管理ページ“リポート（report）”をクライアントブラウザ上の表示のため伝送する。

本発明の一構成によれば、コンテンツサーバーはトランザクションログ５６に記載された記録を定期的に評価して、関連コンテンツサーバーに対するアクセスの頻度とアクセス継続時間を決定する。サーバーは、格付け目的で異なるページ上の情報のメリットを決定するため、共通クライアントからの反復要求を除いて、特定ページに対する要求を計数する。反復要求を除外することで、システムは“投票箱に詰め込む”ことを試みるユーザによるひずみを避ける。一実施形態において、定まった時間周期内に入る共通クライアントによる反復要求を除外するため、共通クライアントによる反復要求の時間間隔が測定される。

これに加え、サーバーは任意の与えられた時間に、クライアント対サーバーのセッション内のアクセス履歴を追跡することができる。このような履歴プロファイルはサービスプロバイダに、リンク横断頻度およびユーザにより辿られるリンク経路について情報を与える。このプロファイルは、特定ユーザＩＤ（ＵＩＤ）を含むトランザクションのみを選択するため、１つまたは複数のサーバーからのトランザクションログをフィルタリングすることで作成される。これらのログにおける特定されたユーザからの要求に対応する２つの連続したエントリであるＡ

およびＢは、この特定されたユーザにより行われたドキュメントＡからドキュメントＢへのリンク横断を表す。この情報は特定ページに対して最も人気があるリンクを識別し、より直接的なアクセスを提供するに何処に新しいリンクを挿入すべきかを示唆するため用いられる。別の実施形態では、商業ページ内で行われた製品の購入につながる横断リンクを決定するためアクセス履歴が評価される。この情報は、例えば、広告ページから製品ページへのリンク横断数、または広告を含む経路から生ずる購入の計数に基づき、広告料の要求に用いることができる。この実施形態では、サーバーは特定ページ、リンク、またはリンクの経路から生じた販売件数を測定することで広告の効果を計ることができる。本システムは、広告ページから生じた販売件数に基づき、広告ページに対する料金を商業者に請求するように構成することができる。

本発明の別の構成によれば、図２Ｂの認証サーバー２００のような第２のサーバーが利用資格データベース２１６から事前に設定されたユーザプロファイルをアクセスすることができ、このようなプロファイルに基づく情報をＳＩＤのユーザ識別子フィールドに含ませることができる。好ましい実施形態では、ＳＩＤのユーザ識別子フィールドに基づき個人化されたコンテンツを含むように、ユーザ要求ページをカスタマイズするためにコンテンツサーバーはこのようなＳＩＤを用いることができる。

本発明の別の構成では、ユーザは予約講読を介して雑誌または刊行物を含むサーバーのドメインに対するアクセスを得ることができる。このような状況では、ユーザはインターネットを介するオンラインドキュメントに対してアクセスを得ることで事前に予約講読権を購入することができる。ユーザは認定識別子が好ましくはセッション識別子に埋設されている上述のような認定手順を介して、インターネット上で予約講読ドキュメントに対するアクセスを得る。別の実施形態では、前払予約講読に頼るのではなく、ユーザがインターネットを介して特定のドキュメントにアクセスする度に料金を請求されることが可能になる。この場合、サービスに対する料金が請求されるためには、ユーザが完全に識別される限り認定は要求されない。ユーザの識別は、最も適切には上記で説明されたセッション

識別子に埋設される。

本発明の別の構成では、商業者のサービスにアクセスするためにユーザが従来の電話番号、またはその他の識別子を利用できるようにする便が設けられている。これらの商業者サービスはＳＩＤを用いて任意に保護され得る。好ましい実施形態では、図６に示されるように、ウェブブラウザクライアント６０１がユーザからの電話番号を受理するため“ダイヤル”アイコン上をクリックし、キーボードを介して電話番号を入力することのような“ダイヤル”コマンドを提供する。ブラウザは次いでＮＵＭＢＥＲがユーザにより指定された電話番号、またはその他の識別

子である、書式“ｈｔｔｐ：／／ｄｉｒｅｃｔｏｒｙ．ｎｅｔ／NUMBER”のＵＲＬを構成する。ブラウザは次いでこのＵＲＬにより指定されたドキュメントのＧＥＴを行い、ディレクトリサーバー６０２に連絡し、メッセージ１で要求されたＮＵＭＢＥＲを送信する。

別の実施形態では、従来のブラウザを備えた形で、クライアント６０１が“ダイヤル”コマンドの場所に電話番号、またはその他の識別子を入れることを促すディレクトリサーバー６０１により設けられた書式ページを用いる。メッセージ１はこの書式ページにより指定されたＵＲＬに対するＰＯＳＴメッセージである。

一旦ＮＵＭＢＥＲがディレクトリサーバー６０１により受信されると、ディレクトリサーバーはデータベース６０４を用いて、ＮＵＭＢＥＲに対応するサービスを実施する商業サーバーおよびドキュメントを記述する目標ＵＲＬにＮＵＭＢＥＲを翻訳する。この翻訳は番号の句読法を無視する可能性があり、従って埋め込まれた括弧またはダッシュは意味を持たない。

別の実施形態では、数字以外の識別子を設けることができる。例えば、ユーザは会社名または製品名を不正確なつづりで記入することがある。このような場合“soundex”または他の音声マッピングが同じ目標ＵＲＬに対するマップと類似して響く語を許容するため用いることができる。また、製品名または内線と組み合わせた電話番号のような複数の識別子も使用可能である。

メッセージ２では、ディレクトリサーバー６０２がＲＥＤＩＲＥＣＴをクライアント６０１に送信し、これには、データベース６０４から演算されたＮＵＭＢＥＲに対する目標ＵＲＬが記述されている。クライアントブラウザ６０１は続いて自動的にメッセージ３を送信し、このＵＲＬのコンテンツをＧＥＴする。商業者サーバー６０３はメッセージ４におけるこの情報を返送する。サーバー６０２はウェブページをクライアントに返送して、要求されたドキュメントに対して適切なリンクを提供

する。しかしながら、サーバー６０２が最終的なＵＲＬに対する翻訳を行い、ページよりはむしろＲＥＤＩＲＥＣＴをクライアント６０１に送信するため、メッセージ４のドキュメントは最初のダイヤル入力以上のユーザの行為なしで入手される。

メッセージ３に含まれる目標ＵＲＬは未管理ページに対する普通のＵＲＬである可能性、または管理ページを記述するＵＲＬである可能性がある。目標ＵＲＬが管理ページを記述する場合、認証は前述のように実行される。また、目標ＵＲＬは管理ページにアクセスする事前認定された手段を提供するＳＩＤを含むＵＲＬを記述できる。

“ダイヤル”コマンドおよびその実施の利点の中には、従来の電話番号および他の識別子と互換性を有する、インターネットにアクセスする改善された方法がある。商業者はコンタクト情報のインターネット特定書式を提供するための印刷物またはテレビ広告を変更する必要がなく、ユーザはＵＲＬについて知る必要がない。

接触時に単独の商業者サーバーが異なる外部の“電話番号”、またはその他の識別子に対応する多数のサービスを提供することができる。例えば、ユーザが“航空便到着”番号にダイヤルした場合、到着ページに対するＵＲＬに導かれる可能性があり、一方、ユーザが“予約”番号にダイヤルした場合、予約ページに対するＵＲＬに導かれるであろう。“優先ゴールド”番号は最初にユーザをゴールドユーザズグループに属するユーザとして立証するであろう管理ページに導かれ、次いで“優先ゴールド”ページに対するアクセスを提供するであろう管理ペー

ジに導かれる可能性がある。未発表の“使節”番号は、ユーザの認証なしに“優先ゴールド”ページに対するアクセスを許すタグ付きＵＲＬに導かれる可能性がある。

本発明は、この明細書に参照として組み入れられた、１９９４年１０月２４日に出願された米国特許出願第０８／３２８，１３３号に提示されたようなネットワーク販売システムに対する特定の用途を有する。

均等物

当業者は、必要以上の実験を行わなくとも、ここに記載された本発明の特定の実施形態に対する多くの均等物を相当するであろう。これらすべての均等物は請求の範囲に包含される。

付属書類

```
/* TclIdSid
 *    Scans an ascii line and finds an ascii SID. (no validation though)
 * Inputs:
 *    lineoftext
 * Returns:
 *    ascii bin_sid, if a sid is found it is returned.
 *
 */

int TclIdSid(ClientData dummy, Tcl_Interp *interp,
                            int argc, char **argv)
    {
    char *sidp, *cp;

    interp->result[0] = 0;

    if (argc != 2)
      {
      interp->result = "wrong # args";
      return TCL_ERROR;
      }
    sidp = (char *) strstr(argv[1], "/@@");
    if (sidp == NULL) return TCL_OK;
    cp = (char *) strstr(sidp+1,"/");
    if ((cp == NULL) && (strlen(sidp) != 19)) return TCL_OK;
    if ((cp - sidp) != 19) return TCL_OK;
    strncpy(interp->result, sidp,19);
    interp->result[19] = 0;
    return TCL_OK;
    }


 /*
  * Register commands with interpreter.
  */
 int SidSupInit(Tcl_Interp *interp)
   {
   Tcl_CreateCommand(interp, "packsid",    TclPackSid,    NULL, NULL);
   Tcl_CreateCommand(interp, "unpacksid", TclUnpackSid, NULL, NULL);
   Tcl_CreateCommand(interp, "unpacksidnovalidate", TclUnpackSidNoValidate,
 NULL,
   Tcl_CreateCommand(interp, "issid",         TclIdSid,        NULL, NULL);
   return TCL_OK;
   }
```

```
/*
 *----------------------------------------------------------------------
 *
 * compute_ihash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 * a 32b xor of the 4 hash longwords.
 *
 * Results:
 *       hash int.
 *
 * Side effects:
 *       None.
 *----------------------------------------------------------------------
 */
int compute_ihash(char *str)
{
    MD5_CTX md5;
    unsigned char hash[16];
    unsigned int *p1;
    unsigned int hashi = 0;

    MD5Init(&md5);
    MD5Update(&md5, str, strlen(str));
    MD5Final(hash, &md5);
    p1 =    (unsigned int *) hash;

    hashi = *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    return hashi;
}

/*
 * ticket.c --
 *
 *       Commands for TICKET.
 *
 * Copyright 1995 by Open Market, Inc.
 * All rights reserved.
 *
 * This file contains proprietary and confidential information and
 * remains the unpublished property of Open Market, Inc. Use,
 * disclosure, or reproduction is prohibited except as permitted by
 * express written license agreement with Open Market, Inc.
 *
```

```
 * Steve Morris
 * morris@OpenMarket.com
 *
 * Created: Wed Mar 1 1995
 * $Source: /omi/proj/master/omhttpd/Attic/ticket.c,v $
 *
 */

#if !defined(lint)
static const char rcsid[]="$Header: /omi/proj/master/omhttpd/Attic/ticket.c,v
2.
#endif /*not lint*/

#include <stdio.h>
#include <sys/utsname.h>
#include "httpd.h"
#include "md5.h"
#include "ticket.h"

static TICKET_Server TicketServerData;

/*
 * This file implements all the ticket/sid related functions for the server.
 *
 * The region commands RequireSID and xxxxx can be used to limit
 * access to groups of files based on the authentication of the requestor.
 * The two commands are very similar, and only differ in the method used to
 * present the authentication data (via the URL) and in handling of the
 * failing access case. For failing TICKET's, a "not authorized" message is
 * generated. For failing (or absent) SID's, a REDIRECT (either local or via
 * CGI script) is performed to forward the request to an authentication
server.
 *
 * RequireSID domain1 [domain2 ... domainn]
 *
 *       This command denies access unless the specified properties are
 *       true of the request. Only one RequireSID or xxxxx command can
 *       be used for a given region, though it may specify multiple domains.
 *
 *
 */

   static int   ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                                int argc, char **argv, int flavor);
   static int   DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                                int argc, char **argv);
   static int   GetDomain(char *domname, int dflt);
```

```
static char *GetAsciiDomain(char *domname, char *dflt);
static int   computer_ihash(char *str);
static char *computerHash(char *str);
static char *GetSecret(int kid);
static int   GetKidByKeyID(char *keyID);
static char *CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid,
                       int exp, int uctx);
static void freeTicketReqData(void *dataPtr);
static void DumpStatus(HTTP_Request *reqPtr);
static void TICKET_DebugHooks(ClientData clientData, char *suffix,
                              HTTP_Request *reqPtr);
static int ParseSid(HTTP_Request *reqPtr);
static int ParseTicket(HTTP_Request *reqPtr);
static char *fieldParse(char *str, char sep, char **endptr);
void TICKET_ConfigCheck();
void DumpRusage(HTTP_Request *reqPtr);

/*
 *----------------------------------------------------------------------
 *
 * TICKET_RequireSidCmd --
 *
 *       Checks that the requested URL is authorized via SID to access this
 *       region. If the access is not authorized and we do not have a "remote"
 *       authentication server" registered, then an "unauthroized message"
 *       is returned.  If a "remote authentication server" has been
 *       declared, we REDIRECT to that server, passing the requested URL and
 *       required domain's as arguments.
 *
 * Results:
 *       Normal Tcl result, or a REDIRECT request.
 *
 * Side effects:
 *       Either an "unauthorized access" message or a REDIRECT in case of
error.
 *
 *----------------------------------------------------------------------
 */
static int TICKET_RequireSidCmd(ClientData clientData, Tcl_Interp *interp,
                                int argc, char **argv)
    {
    if (TicketGlobalData(EnableSidEater)) return TCL_OK;
    return(ProcessRequires(clientData, interp,argc, argv, ticketSid));
    }

/*
 *----------------------------------------------------------------------
 *
```

```
* ProcessRequired --
*
*       Checks that the requested URL is authorized to access this
*       region. The error cases are treated differently for SID v.s. TICKET.
*       For Ticket's, an unauthorized access generates a returned error
message.
*       For SID's, we first look to see if we are operating in "local
authentica
*       mode", if we are, we generate a new SID, into the URL and re-process
the
*       If not in "local" mode, we look for the presence of a
remoteauthenticati
*       server, if we have one declared (in the conf file) we REDIRECT to it
pas
*       the FULL url and a list of domains that would have been legal.  If
the
*       authentication server was not found we return an error message.
*
* Results:
*       Normal Tcl result, a local reprocess command, or a REDIRECT request.
*
* Side effects:
*       Either an "unauthorized access" message or a REDIRECT in case of
error.
*
*----------------------------------------------------------------------
*/
static int ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                                    int argc, char **argv, int flavor)
    {
    HTPP_Request *reqPtr = (HTTP_Reqeust *) client Data;
    HTTP_Server  *serverPtr;
    TICKET_Request *ticketPtr;
    DString targetUrl;
    DString escapeUrl;
    int i, required_dom;
    int firstLegalDom = -1;
    char *NewSid, *cp;

    DStringInit(&targetUrl);
    DStringInit(&escapeUrl);

    /* fetch the server private and ticket specific extension data */
    serverPtr = reqPtr=>serverPtr;
    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr,
 TicketServerData.tic
    ASSERT (ticketPtr != NULL);
```

```
    /* compare the requesting SID/Ticket<DOM> to authorized list of domains */
    /* a match OR any valid domain and a required domain of TicketFreeArea is
su
   for (i = 1; i < argc; i++)
     {
     required_dom = GetDomain(argv[i] -1);
     if (required_dom != -1)
       {
       if (firstLegalDom == -1) firstLegalDom = required_dom;
       if ( (ticketPtr->sidDom == required_dom) ||
            (ticketPtr->valid && (ticketPtr->sidDom != -1) &&
              (required_dom == TicketGlobalData(FreeArea))) ||
            ((ticketPtr->ticketDom == required_dom) &&
              (time(0) <= ticketPtr->ticketExp) &&
              ((DStringLength(&ticketPtr->ticketIP) == 0) ||
              (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr-
>r
          )
            {
            DStringFree(&targetUrl);
            DStringFree(&escapeUrl);
            return TCL_OK;
            }
     }
    }

 /* count the number of domain crossing that caused re-auth */
 if ((flavor == ticketSid) && (ticketPtr->sidDom) != -1) IncTicketCounter(Cou

 /* authorization failed, if this was a sid url, and local auth is enabled */
 /* or this was an access to the free area */
 /* insert a new sid in the url, and REDIRECT back to the client 8?
 if (TicketGlobalData(EnableLocalAuth) ||
      ((firstLegalDom == TicketGlobalData(FreeArea))
      && (flavor == ticketSid) && (firstLegalDom != -1)))
    {
    if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url)[0] != '/'))
      {
      HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
      DStringFree(&targetUrl);
      DStringFree(&escapeUrl);
      if (!ticketPtr->valid)
        DStringFree(&ticketPtr->sid);
      return TCL_RETURN;
      }
    NewSid =    CreateSid(reqPtr,
```

```
            firstLegalDom, ticketPtr->uid,
            TicketGlobalData(CurrentSecret), TicketGlobalData(LocalAuthExp)
            ticketPtr->uctx);
  DStringFree(&ticketPtr->sid);
  DStringAppend(&ticketPtr->sid, NewSid, -1);
  ComposeURL(reqPtr, DStringValue(&reqPtr->url), &targetUrl);
  IncTicketCounter(CountLocal Redirects);
  HTTP_Error*reqPtr, REDIRECT, DStringValue(&targetUrl));
  DStringFree(&targetUrl);
  DStringFree(&escapeUrl);
  if (!ticketPtr->valid)
    DStringFree(&ticketPtr->sid);
  return TCL_RETURN;
  }

/* authorization failed, build the REDIRECT URL arg's. */
/* If present, REDIRECT to authentication server */
if ((DStringLength(&TicketGlobalData(AuthServer)) != 0) && (flavor == ticket
  {
  if ((DStringLength(&reqPtr->url) != 0) &&
      (DStringValue(&reqPtr->url) [0] != '/'))
    {
    HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
    DStringFree(&targetUrl);
    DStringFree(&escapeUrl);
    if (!ticketPtr->valid)
      DStringFree(&ticketPtr->sid);

  return TCL_RETURN ;
  }
DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(AuthServer)), -1)
DStringAppend(&targetUrl, "?url=", -1);
ComposeURL(reqPtr, DStringValue(&reqPtr->url), &escapeUrl)
EscapeUrl(&escapeUrl);
DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
DStringAppeal(&targetUrl, "&domain=", -1);
DStringTrunc(&escapeUrl, 0);
DStringAppend(&escapeUrl, "{=, -1);
for (i=1; i < argc; i++)
  [
  cp = GetAsciiDomain*argv[i], NULL);
  if (cp != NULL)
    {
    DStringAppend(&escapeUrl, cp, -1);
    DStringAppend(&escapeUrl, " ", -1);
    }
  }
```

```
DStringAppend(&escapeUrl, "}", -1);
EscapeUrl(&escapeUrl);
DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
DStringFree(&escapeUrl);
HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
IncTicketCounter(CountRemoteRedirects);
DStringFree(&targetUrl);
if (!ticketPtr->valid)
  DStringFree(&ticketPtr->sid);
return TCL_RETURN;
}

/* authorization failed, if this is a ticket access, decode the */
/* reason and handl via a redirect to a handler, or punt a      */
/* no access message */
if ((flavor == ticketTicket) && (firstLegalDom != -1) && (ticketPtr->ticketD
  {
  /* check For IP address restrictions */
  if ((DStringLength(&ticketPtr->ticket IP) != 0) &&
      (DstringLength(&TicketGlobalData(TicketAdrHandler)) != 0) &&
      (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr->remo
    {
    DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(TicketAdrHandle
    DStringAppend(&targetUrl, DStringValue(&ticketPtr->fields), -1);
    DStringAppend(&targetUrl, "&url=", -1);
    DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
    IncTicketCounter(CountTicketAddr);
    HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
    DStringFree(&targetUrl);
    return TCL_RETURN;
    }

/* check for expired tickets */
if (time(0) > ticketPtr->ticketExp)
   {
   DStringAppend(&targetUrl,    DStringValue(&TicketGlobalData(TicketExpHandle
   DStringAppend(&targetUrl,    DStringValue(&ticketPtr->fields), -1);
   DStringAppend(&targetUrl, "&url=", -1);
   DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
   IncTicketCounter(CountExpiredTicket);/*

HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
DStringFree(&targetUrl);
return TCL_RETURN;
   }
}
```

```
/* no handler, punt a message */
HTTP_Error(reqPtr, FORBIDDEN, "access denied by Require ticket/sid region
co
IncTicketCounter(CountNoRedirects);
if (!ticketPtr->valid)
DStringFree(&ticketPtr->sid);
DStringFree(&targetUrl);
DStringFree(&escapeUrl);
return TCL_RETURN;
}

/*
*----------------------------------------------------------------------
*
* Get(Ascii)Domain --
*  These routine performs an ascii to binary domain name lookup,
*  indexed by 'key') from the server's domain name catalog.  Name/number
*  pair's are loaded into the catalog at configuration time with the
*  with the "Domain" configuration command. The Ascii version returns
*  a pointer to a character string that represents the domain number.
*  The non Ascii version returns an integer representing the domain number.
*
* Results:
*  Integer value of domain.  If no domain is available, returns deflt.
*
* Side effects:
*  None.
*
*----------------------------------------------------------------------
*/
static int GetDomain (char *domname, int deflt)
   {
   HashEntry *entryPtr;
   DString DomName;

   DStringInit(&DomName);
   DStringAppend(&DomName, domname, -1);
   strtolower(DStringValue(&DomName));

   entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
   DStringFree(&DomName);
   if (entryPtr == NULL) return deflt;
```

```
  return (int) GetHashValue(entryPtr);
  }
static char * GetAsciiDomain(char *domname, char *deflt)
  {
  HashEntry *entryPtr;
  static char buffer[64];
  DString DomName;

  DStringInit (&DomName);
  DStringAppend(DomName, domname, -1);
  strtolower(DStringValue(&DomName));

  entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
  DStringFree(&DomName);
  if (entryPtr == NULL) return deflt;
  sprintf(buffer, "%d", (int) GetHashValue(entryPtr));
  return buffer;
  }
/*
*----------------------------------------------------------------------
*
*  TICKET_InsertLocalSid --
*
* Given a URL, inspect it to see if it refers to the local server/port
* if it does, and it does not already contain a SID, insert one if
* the current request included one. Note, for port 80 access we look
* for a match with and without the port specifier.
*
* Results:
* None.
*
* Side effects:
  A SID may be inserted into the URL.
*
*----------------------------------------------------------------------
*/
void TICKET_InsertLocalSid(HTTP_Request *reqPtr, DString *result)
  {
  HTTP_Server *serverPtr;
  TICKET_Request *ticketPtr;
  char tmp[32];
  DString pattern1;
```

```
  DString pattern2;
  DString tmp_url;
  DString *hitPattern * NULL;

  ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr,
TicketServerData.tic
  if (ticketPtr == NULL) return;
  serverPtr = reqPtr->serverPtr;

  DStringInit(&pattern1);
  DStringInit(&pattern2);
  DStringInit(&tmp_url);

  DStringAppend(&pattern1, "http://", -1);
  DStringAppend(&pattern1, DStringValue(&serverPtr->serverName), -1);
  DStringAppend(&pattern2, DStringValue (&pattern1), -1);
  sprintf(tmp, ":%d", serverPtr->server_port);
  DStringAppend(&pattern1, tmp, -1);

  if ((DStringLength(result) >= DStringLength(&pattern1)) &&
   (strncasecmp(DStringValue(&pattern1), DStringValue(result),
DStringLengt hitPattern = &pattern1;
else
if  ((serverPTR-->server_port == 80) &&
  (DStringLength(result) >= DStringLength(&pattern2)) &&
  (strncasecmp(DStringValue(&pattern2), DStringValue(result),
DStringLength hitPattern + &pattern2;

  if (hitPattern ! = NULL)
  {
  DStringAppend(&tmp_url, DStringValue(hitPattern), -1;
  DStringAppend(tmp_url, DStringValue(&ticketPtr->sid), -1);
  DStringAppend(&tmp url, &DStringValue(result)
[DStringLength(hitPattern)],
  DStringFree(result);

  DStringAppend(result, DStringValue(&tmp_url), -1):
  DStringFree(&tmp_url);
  }

  DStringFree(&pattern1);
  DStringFree(&pattern2);
  DStringFree(&tmp_url);
  }
 /*
```

```
*----------------------------------------------------------------------
*
* CreateSid --
*
  This routine takes the passed arguments and creates a sid.
*
* Results:
* A sid.
*
* Side effects:
*
*----------------------------------------------------------------------
*/
char * CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid, int exp,
int uctx)
   {
   int bsid[3] = {0,0,0};
   char temp_str[512];
   DString hash;
   int act_hash;
   static char sid[64];
   unsigned int expire_time;
   char *secret;)
   char *hashP;
   char *cp;
   unsigned char *ecp;
   unsigned int eda;
   int endian = 1;

   DStringInit(&hash);
   expire_time =time(0)+ exp;

   put_sid(dom_lw,       dom_pos,    dom_mask,   dom);
   put_sid(uid_lw,       uid_pos,    uid_mask,   uid);
   put_sid(kid_lw,       kid_pos,    kid_mask,   kid);
   put_sid(exp_lw,       esp_pos,    exp_mask,
 (expire_time>>exp_shft_amt))
   put_sid(uctx_lw,      uctx_pos,   uctx_mask,  uctx);
   put_sid(rev_lw,       rev_pos,    rev_mask,   sid_rev_zero);

   secret = GetSecret(kid);
   ASSERT (secret ! = NULL);
   DStringAppend(&hash, secret, -1);
```

```
  DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1;
   sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
   DStringAppend(&hash, temp_str, -1);
   /* format of the hash string is %s%s%08x%08x",
secret,ip_addr,bsid[2[,bsid[1
   hashP = DStringValue (&hash);
   act_hash = compute_ihash(hashP);
   while (*hashP ! = 0) *hashP++ = 0;
   DStringFree(&hash);
/*  fix_endian(&act_hash, ecp, eda); */

    .put_sid(sig_lw, sig_pos, sig_mask, act_hash)

/*  fix_endian(&bsid[0], ecp, eda); */
     fix_endian(&bsid[1], ecp, eda);
     fix_endian*&bsid[2], ecp, eda);

#if (1 == 0
   DumpSid();
#endif

     cp = radix64encode_noslash((char *) bsid, 12);
     strcpy(sid, SID_prefix);
     strcat(sid, cp);
     free(cp);
     return(sid);
     }

  /*
   *----------------------------------------------------------------------
   *
   *   compute_hash --
   *
   *   Compute the MD5 hash for the specified string, returning the hash as
   *   a 32 b xor of the 4 hash longwords.
   *
   *  Results:
   hash int.
   *

   *  Side effects:
   None.
  *----------------------------------------------------------------------
```

```
 */
static int compute_ihash(char *str)
  {
   MD5_CTX md5;
   unsigned char hash[16];
   unsigned int *pl;
   unsigned int hashi = 0;

  MDInit(&md5);
  MDUpdate(&md5, (unsigned char *) str, strlen(str));
  MDFinal(hash, &md5);
  pl = (unsigned int *) hash;

  hashi = *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  return hashi;
  }

/*
 *----------------------------------------------------------------------
 *
 *  computeHash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 *  a 32-character hex string.
 *
 *  Results:
  Pointer to static hash string.
 *
 * Side Effects:
 *  None.
 *----------------------------------------------------------------------
 */
static char *computeHash(char *str)
{
  int i;
  MD5_CTX md5;
  unsigned char hash[16];
  static char hashstr[33];
  char *q;
```

```
  MD5Init(&md5);
  MD5Update(&md5, (unsigned char *) str, strlen(str));
  MD5Final(hash, &md5);
  q = hashstr;
  for(i=0; i<16; i++ {
   sprintf(q, "%02x", hash[i]);
   q += 2;
  }
  *q = '\0';
  return hashstr;
}

/*
 *----------------------------------------------------------------------
 *
 *  TICKET_ParseTicket --
 *  Called by dorequest, before any region commands or mount handlers
 *  have run.  We parse and handle incomeing sid's and tickets.
 *
 * Results:
 *  None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------

 */
int TICKET_ParseTicket(HTTP_Request *reqPtr)
   {
   int status = HT_OK;

   IncTicketCounter(CountTotalUrl);

   status = ParseSid(reqPtr);
   if (TicketGlobalData(EnableTicket) && (status == HT_OK)) status =
ParseTicke return status;
   }

 /*
 *----------------------------------------------------------------------
 *
 * ParseSid --
 *
```

```
* Called by TICKET_ParseTicket, before any region commands or mount handle
* have run.  We parse and handle incomeing sid's.
*
* Results:
*  None.
*
* Side effects:
*
*----------------------------------------------------------------------------
*/

int ParseSid(HTTP_Request *reqPtr)
   {
   TICKEKT_Request *ticketPtr;
   HTTP_Server *serverPtr;
   DString hash;
   Int i;
   char *cp, *cp1;
   int *bsid=NULL, act_hash;
   unsigned int cur_tim, tdif, exp_tim;
   char *secret;
   char temp_str[512];
   char *hashP;
   int sid_ok = 0;
   unsigned char *ecp;
   unsigned int eda;
   int endian = 1;
   int ip1,ip2,ip3,ip4;

/* fetch the server private ticket extension data */
/* note that this sets up a default ticket block for both SID's and Ticket a
serverPtr = reqPtr->serverPtr;
ticketPtr = (TICKET_Request *) HT_GetReqExtData (reqPtr, TicketServerData.tic
ASSERT (ticketPtr == NULL);

ticketPtr = (TICKET_Request *) Malloc(sizeof(TICKET_Request));
HT_AddReqExtData(reqPtr, TicketServerData.ticketExtensionId, ticketPtr, free
DStringInit(&ticketPtr->rawUrl);
DStringInit(&ticketPtr->sid);
DStringInit(&ticketPtr->fields);
DStringInit(&TicketPtr->signature);
DStringInit(&TicketPtr->ticketIP);
ticketPtr->valid      =  0;
ticketPtr->sidDom     =   -1;
ticketPtr->ticketDom  =  -1;
ticketPtr->ticketExp  =  -1;
ticketPtr->uid        =  0
```

```
TicketPtr->uctx        =  0;
sscanf(DStringValue(&reqPtr->remoteAddr), "%d.%d.%d.%d", &ip1, &ip2, &ip3, &
ticketPtr->uid = (((ip1+ip2)<<24) | ((ip3+ip4)<<16) | (rand() & 0xFFFF));
ticketPtr->uctx  = 1;
/* we are done if sids are not enabled, or this url does not have a sid */
if (!(TicketGlobalData(EnableSid))) return HT_OK;
cp1 = DStringValue(&reqPtr->url;
if (strstr(cp1, SID prefix) ! = cp1)
  return HT_OK;
if (strlen(cp1) == sidLength)
  {
  DStringAppend(&reqPtr->url, "/", -1);
  DStringAppend(&reqPtr->path, "/", -1);
  cp1 = DStringValue(&reqPtr->url);
  }
cp = strchr(cp1+sizeof(SID_prefix),'/');
if ((cp - cp1) != sidLength)
  return HT_OK;
IncTicketCounter(CountSidUrl);

DStringInit(&hash);

/* if sid eater is enabled, rewrite the url without the sid, and reprocess t
if (TicketGlobalDat(EnableSidEater))
  {
  DStringAppend(&hash, DStringValue(&reqPtr->url), -1);
  DStringFree(reqPtr->url);
  DStringAppend(&reqPtr->url, DStringValue(&hash)&hash)+sidLength, -1);
  DStringTrunc(&hash, 0);
  DStringAppend(&hash, DStringValue(&reqPtr-.path), -1);
  DStringFree(&reqPtr->path);
  DStringAppend(&reqPtr->path, DStringValue(&hash)+sidLength, -1);
  DStringFree(&hash);
  IncTicketCounter(CountDiscardedSidUrl);
  return HT_OK;
  }

DStringAppend(&ticketPtr->sid,  DStringValue(&reqPtr->url), sidLength);

/* first convert the SID back to binary*/
i = DStringLength(&ticketPtr->sid)-3;
bsid = (int *) radix64decode_noslash(DStringValue(&ticketPtr->sid)+3, i, &i)
iif ((bsid == NULL) || (i !+12)) goto rtn_exit;

fix_endian(&bsid[0], ecp, eda);
fix_endian(&bsid[1], ecp, eda);
fix_endian(&bsid[2], ecp, eda);
```

```
/* check the SID version field */
if (get_sid(rev_lw,rev_pos,rev_mask) ! = sid_rev_zero) goto sid_bad;
if (get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask) ! = 0)) goto sid_bad;
if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) !+ 0) goto sid_bad;

/* Get a pointer to the secret */
secret = GetSecret(get_sid(kid_lw,kid_pos,kid_mask));
if (secret == NULL) goto sid_bad;

/* hash the sid and check the signature*/
DStringAppend(&hash, secret, -1);

DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1);
sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
dstringAppend(&hash, temp_str, -1);
/* format of the hash string is %s%s%08x%08x", secret,ip_addr,bsid[2],bsid[1

hashP = DStringValue(&hash);
act_hash = compute_ihash(hashP);
while (*hashP != 0) *hashP== 0;
fix_endian(&act_hash, ecp, eda);
if (act_hash != get_sid(sig_lw,sig_pos,sig_mask)) goto  sid_bad;

/* is is ok, may be expired, but good enough to id user  */
ticketPtr->uiid = get_sid(uid_llw,uid_pos,uid_mask);
ticketPtr->uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

/* do the SID experation processing*/
cur_tim = (time(0)>>exp_shft_amt) & exp_mask;
expp_tim = get_sid(exp_lw,exp_pos,exp_mask);
tdif = (exp_tim - cur_tim) & 0xffff;
if (tdif > 0X7fff)
  {
  IncTicketCounter(countExpSid);
  goto sid_exp;
  }

/* sid is fine, save the sid state, update the url's */
ticketPtr->sidDom = get_sid(dom_lw,dom_pos,dom_mask);
ticketPtr->valid = 1;
sid_ok = 1;
IncTicketCounter(CountValidSid);

sid_bad:
  if (!(sid_ok)) IncTicketCounter(CountInvalidSid);
sid_exp:
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->path), -1);
```

```
  DStringTrunc(&reqPtr->path, 0);
  DStringAppend(&reqPtr->path, DStringValue(&ticketPtr->rawUrl)+sidLength, -1)

DStringTrunc(&ticketPtr->rawUrl, 0);
DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->url), -1);
DStringTrunc(&reqPtr->url, 0);
DStringAppend(&reqPtr->url, DStringValue(&ticketPtr->rawUrl)+sidLength, -1);

rtn_exit:
  DStringFree(&hash);
  if (bsid != NULL) free(bsid);
  return HT_OK;
  }

/*
 *----------------------------------------------------------------------
 *
 * freeTicketReqData
 *
 *  This routine frees the storage used by ticket specific request
 *  data.
 *
 * Results:
 *  None.
 *
 * Side effects:
 *  Memory freed.
 *
 *----------------------------------------------------------------------
*/

;tatic void freeTicketReqData(void *dataPtr)
   {
   TICKET_Request *ticketPtr = dataPtr;
   DStringFree(&ticketPtr->rawUrl);
   DStringFree(&ticketPtr->sid);
   DStringFree(&ticketPtr->fields);
   DStringFree(&ticketPtr->signature);
   DStringFree(&ticketPtr->ticketIP);
   free(ticketPtr);
   }
/*
 *----------------------------------------------------------------------
 *
 * GetSecret --
 *
 *  Given a binary keyID, returns an ascii secret from the
```

```
 *  secrets store.
 *  for untranslatable names, return NULL.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

char *GetSecret(int kid)
  {
  HashEntry *entryPtr;

  entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);
  if(entryPtr == NULL) return NULL;
  return DStringValue((DString *)GetHashValue(entryPtr));
  }


/*
 *----------------------------------------------------------------------
 *
 * GetKidByKeyID --
 *
 *  Given an ascii  KeyID return the binary Key ID.
 *  for untranslatable names, return -1.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

int GetKidByKeyID(char *keyID)
  {
  HashEntry *entryPtr;

  entryPtr = FindHashEntry(*&TicketServerData.KeyID< (void *) keyID);
  if(entryPtr == NULL) return -1;
  return (int) GetHashValue(entryPtr);
  }
```

```
/*
 *----------------------------------------------------------------------
 *
 * fieldParse --
 *
 *  Given a string, a separator character, extracts a field up to the
 *  separator into the result string.
 *  Does substitution on '%XX' sequences, and returns the pointer to the
 *  character beyond last character in '*endptr'.
 *
 * Results:
 *  Returns a malloc'ed string (caller must free), or NULL if an
 *  error occurred during processing (such as an invalid '%' sequence).
 *
 * Side effects:
 *  None.
 *
 *----------------------------------------------------------------------
*/
#define SIZE_INC 200
statiic char *fieldParse(char *str, char sep, char **endptr)
{
   char buf[3];
   char c;
   char *end, *data, *p;
   int maxlen, len;

   len = 0;
   maxlen = SIZE_INC;
   p = data = malloc(maxlen);

 /*
  * Loop through string, until end of string or sep character.
  */
  while(*str && *str != sep) {

    if(*str == '%') {

      if(!isxdigit(str([1]) || !isxdigit(str[2])) {
           free(data);
           return NULL;
      }
      buf[0] = str [1];
      buf[1] = str [2];
      buf[2] = '\0';
      c = strtol(buf, &end, 16);
      str += 3;
```

```
  } else if(*str == '+') {
   c = ' ';
   str++;
  } else
   c = *str++;

  *p++ = c;
  len++;
  if(len >= maxlen) {
   maxlen += SIZE_INC;
   data = realloc(data, maxlen);
   p = data + len;
  }


   }
   *p++ = '\0';
   *endptr = str;
   return data;
  }
/*
 *----------------------------------------------------------------------
 *
 *  DomainNameCmd --
 *
 *  A call to this routine, builds the ascii domain name
 *  to binary domain name maping structure for a numeric domain.
 *  Syntax is Domain number name1 name2 name3 name...name_last
 *  At least one name is required.  The number is decimal and
 *  can be any value except -1. -1 is reserved as a marker
 *  for untranslatable names.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Commands are validate, and entries added to the map
 *
 *----------------------------------------------------------------------
 */
static int DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                         int argc, char **argv)
   {
   int new,i;
   HashEntry *entryPtr;
   int DomNumber;
   DString DomName;
```

```
if (argc <3)
   {
   Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
                "arguments, should be \"3\"",
        (char *) NULL);
   return TCL_ERROR;
   }

DStringInit (&DomName);

if (((sscanf(argv[1], "%d", &DomNumber) ! = 1 || (DomNumber == -1)))
   {
   Tcl_AppendResult(interp, argv[0], " directive: ",
            "Domain number must be an integer, and not equal to -1",
            ", value found was ",argv[1],
            (char *) NULL);
   return to TCL_ERROR;
   }
for (i = 2; i < argc; i++)
   {
   DStringFree (&DomName);
   DStringAppend(&DomName, argv[i], -1);
   strtolower(DString Value(&DomName));
   entryPtr = CreateHashEntry(&TicketServerData.Domains, DStringValue
     (&DomNam
   if (new == 0)
   {
   Tcl_AppendResult(interp, argv[0], " directive:

    "Duplicate domain name specified, '", argv[i], "'",
    (char *) NULL);
   return TCL_ERROR;
   }
    SetHashValue(entryPtr, DomNumber);
    }
DStringFree(&DomName);
return TCL_OK;
}

/*
 *----------------------------------------------------------------------
 *
 *   SecretsCmd --
 *
 *  A call to this routine, builds kid to secrets table
 *
 *  Results:
```

```
 *  None.
 *
 *  Side effects:
 *  Secrets are stored.
 *
 *----------------------------------------------------------------------
 */
static int SecretsCmd(ClientData clientDate, Tcl_Interp *interp,
                int argc, char **argv)
   {
   int newKid,newKeyID;
   HashEntry *entryPtrKid = NULL, *entryPtrKeyID = NULL;
   int Kid;
   DString *dsptrKid;

   if (argc ! = 4)
    {
  Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
        "arguments, should be \"4\" ",
        (char *) NULL);
   return TCL_ERROR;
   }

   if (sscanf(argv[2], "%d", &Kid) ! = 1)
   {
   Tcl_AppendResult(interp, argv[0],
        " directive:  KeyID must be an integer",
        ", value found was '", argv[2], "'",
        (char *) NULL);
   return TCL_ERROR;
   }

  entryPtrKid = CreateHashEntry(&TicketServerData.SecretsKid, (void *) Kid, &n
  if (strlen(argv[1]))
    entryPtrKeyID = CreateHashEntry(&TicketServerData.KeyID, (void *) argv[1],
  if ((newKid == 0 || ((newKeyID == 0) && strlen(argv[1]))))
   {
   Tcl_AppendResult(interp, argv[0],
        " directive:  Duplicate Secret specified for KeyID '",
        argv[1],
        (char *) NULL);
   return TCL_ERROR;
   }
   if (strlen(argv[1]))
   {
   dsptrKid = (DString *) malloc(sizeof(DString));
   DStringInit(dsptrKid);
```

```
  DStringAppend(dsptrKid, argv[3], -1);

  SetHashValue(entryPtrKid, dsptrKid);
  }
  SetHashValue(entryPtrKeyID, Kid);
  return TCL_OK;
  }


/*
 *----------------------------------------------------------------------
 *
 *  TICKET_Initialize --
 *
 *  Calls all the necessary routines to initialize the ticket subsystem.
 *
 *  Results:
 *       None.
 *
 *  Side effects:
 *       Commands added to the region interpreter.
 *  SID "/@@" url catcher declared.
 *
 *----------------------------------------------------------------------
 */
int TICKET_Initialize(HTTP_Server &serverPtr, Tcl_Interp *interp)
   {
   TicketServerData.ticketExtensionId = HT_RegisterExtension(serverPtr,
   "ticket

   InitHashTable(&TicketServerData.SecretsKid, TCL_ONE_WORD_KEYS);
   InitHashTable(&TicketServerdata.KeyID,    TCL_STRING_KEYS);
   InitHashTable(&TicketServerData.Domains, TCL_STRING_KEYS);

   /* initialize Server ticket data */
   DStringInit(&TicketGlobalData(AuthServer));
   DStringInit(&TicketGlobalData(TicketExpHandler));
   DStringInit(&TicketGlobalData(TicketAdrHandler));
   TicketGlobalData(FreeArea)                = 0;
   TicketGlobalData(EnableLocalAuth)         = 0;
   TicketGlobalData(CurrentSecret)           = 0;
   TicketGlobalData(EnableSid)               = 0;
   TicketGlobalData(EnableTicket)            = 0;
   TicketGlobalData(EnableSidEater)          = 0;
   TicketGlobalData(LocalAuthExp)            = 60*30;

   /* ticket event counters */
```

```
TicketGlobalData(CountTotalUrl)          = 0;
TicketGlobalData(CountSidUrl)            = 0;
TicketGlobalData(CountValidSid)          = 0;
TicketGlobalData(CountExpSid)            = 0;
TicketGlobalData(CountInvalidSid)        = 0;
TicketGlobalData(CountCrossDomain)       = 0;
TicketGlobalData(CountLocalredirects)    = 0;
TicketGlobalData(CountRemoteRedirects)   = 0;
TicketGlobalData(CountNoRedirects)       = 0;
TicketGlobalData(CountDiscardedSidUrl)   = 0;

/* Ticket related Config commands */
Tcl_CreateCommand(interp, "Domain",                    DomainNameCmd,
          (ClientData) serverPtr, NULL);
Tcl_CreateCommand(interp, "Secrets",           SecretsCmd,
          (ClientData) serverPtr, NULL);
Tcl_CreateCommand(interp, "AuthenticationServer", CmdStringValue,
       (ClientData) &TicketGlobalData(AuthServer), NULL);
Tcl_CreateCommand(interp, "TicketExpirationHandler", CmdStringValue,
  (ClientData) &TicketGlobalData(TicketExpHandler), NULL);
Tcl_CreateCommand(interp, "TicketAddressHandler", CmdStringValue,
  (ClientData) &TicketGlobalData(TicketAdrHandler), NULL);
Tcl_CreateCommand(interp, "FreeDomain",         CmdIntValue,
  (ClientData) &TicketGlobalData(FreeArea), NULL);
Tcl_CreateCommand(interp, "EnableSidEater",     CmdIntValue,
  (ClientData) &TicketGlobalData(EnableSidEater), NULL);
Tcl_CreateCommand(interp, "EnableSid",          CmdIntValue,
  (ClientData) &TicketGlobalData(EnableSid), NULL);
Tcl_CreateCommand(interp, "EnableTicket",       CmdIntValue,
  (ClientData) &TicketGlobalData(EnableTicket), NULL);
Tcl_CreateCommand(interp, "EnableLocalAuth",    CmdIntValue,
  (ClientData) &TicketGlobalData(EnableLocalAuth), NULL);
Tcl_CreateCommand(interp, "CurrentSecret",      CmdIntValue,
  (ClientData) &TicketGlobalData(CurrentSecret), NULL);
Tcl_CreateCommand(interp, "LocalAuthExp",       CmdIntValue,
  (ClientData) &TicketGlobalData(LocalAuthExp), NULL);

HT_AddMounthandler(serverPtr, (ClientData) NULL, TICKET_DebugHooks,
  "/omiserver", NULL);

return HT_OK;
}


/*
 *----------------------------------------------------------------------
 *
```

```
 * TICKET_Shutdown --
 *
 *  Calls all the necessary routines to shutdown the ticket subsystem.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Memory freed
 *
 *----------------------------------------------------------------------
 */

void TICKET_Shutdown (HTTP_Server *serverPtr)
   {
   HashEntry *entryPtr;
   HashSearch search;
   DString   *dstring;

   DStringFree(&TicketGlobalData(AuthServer));
   DStringFree(&TicketGlobalData(TicketExpHandler));
   DStringFree(&TicketGlobalData(TicketAdrHandler));

   entryPtr = FirstHashEntry(&TicketServerData.SecretsKid, &search);
   while (entryPtr != NULL)
     {
     dstring = GetHashValue(entryPtr);
     DStringFree(dstring);
     free(dstring);
     entryPtr = NextHashEntry&search);
     }
   DeleteHashTable(&TicketServerData.SecretsKid);
   DeleteHashtable(&TicketServerData.KeyID);
   DeleteHashTable(&TicketServerData.Domains);
   }
/*
 *
----------------------------------------------------------------------
-
 *
 * TICKET_AddRegion Commands --
 *
 *          Add TICKET region commands for authentication/authorization
decisions.
 *
 * Results:
            None.
```

```
 *
 * Side effects:
          Commands added to the region interpreter.
 *
 *----------------------------------------------------------------------
 */

void TICKET_AddRegion Commands (HTTP_Request *reqPtr, Tcl_Interp *interp)
      {
      Tcl_CreateCommand(interp, "RequireSID", TICKET_RequireSidCmd,
                        (ClientData) reqPtr, NULL);
      Tcl_CreateCommand(interp, "RequireTicket", TICKET_RequireTicketCmd,
                        (ClientData) reqPtr, NULL);
      }


/*
 *----------------------------------------------------------------------
 *
 * TICKET_GetCGIVariables --
 *
 *    Add TICKET CGI variables to the CGI variable table.
 *
 * Results:
          None.
 *
 * Side effects:
          Extends the CGI variable hash table.
 *
 *----------------------------------------------------------------------
 */

void TICKET GetCGIVariables(HTTP_Request *req)
      {
      TICKET_Request *ticketPtr = (TICKET_Request *)
HT_GetReqExtData(req.TicketS

      /*
       * If there's no extension data, then we're not doing a ticket.  Just
return
       */

      if (ticketPtr == NULL)
         return)\;
```

```
      if  (DStringLength(&ticketPtr->rawUrl) ! = 0)
           HT_AddCGIParameter(req, "TICKET_URL", DStringValue(&ticketPtr-
>rawUrl), FA
      if (DStringLength (&ticketPtr->sid) != 0)
           HT_AddCGIParameter(req, "TICKET_SID", DStringValue(&ticketPtr-
>sid), FALSE
      if (DStringLength(&ticketPtr->fields) != 0)
           HT_AddCGIParameter(req, "TICKET_FIELDS", DStringValue(&ticketPtr-
>fields).
      if (DStringLength(&ticketPtr->signature) != 0)
      HT-AddCGIParameter(req, "TICKET_SIGNATURE". DStringValue(&ticketPtr-
>signa
      }/*
 *---------------------------------------------------------------------------
 *
 *TICKET_GetUrl
 *
 *          Return the orignal url (with sid)
 *
 * Results:
 *          The URL.
 *
 * Side effects:
           None.
 *
 *---------------------------------------------------------------------------
 */
char * TICKET_GetUrl(HTTP_Request *reqPtr)
      {
      TICKET_Request *ticketPtr;

      ticketPtr = (TICKET_Request *)
           HT_GetReqExtData(reqPtr, TicketServerData.ticketExtensionId);
      if ((ticketPtr != NULL) &&
            (DStringLength(&ticketPtr->rawUrl) != 0))
           return DStringValue(&ticketPtr->rawUrl);
      else
           return DStringValue(&reqPtr->url);
      }


 /*
  *---------------------------------------------------------------------------
  *
  * TICKET_ConfigCheck
  *
  *          Perform late configuration checks
```

```
 *
 * Results:
 *
 *
 * Side effects:
 *          Possible message loged/printed, and program exit'd.
 *
 *----------------------------------------------------------------------
 */
void TICKET_ConfigCheck()
       {
       HashEntry *entryPtr;
       int kid;

       if ((TicketGlobalData(EnableSid) & -0x1) != 0)
            {
            LogMessage(LOG_ERR, "EnableSid must be 0 or 1");
            exit (0);
            }
       if (!(TicketGlobalData(EnableSid))) return;

       kid = TicketGlobalData(CurrentSecret);
       if (kid && kid_mask) != kid)
            {
            LogMessage(LOG-ERR; "CurrentSecret %d is invalid", kid);
            exit(0);
            }

       entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);

       if(entryPtr == NULL)
            {
            LogMessage(LOG_ERR), "No secret defined for CurrentSecret %d", kid;
            exit(0);

       if ((TicketGlobalData(FreeArea) & -0x255) != 0)
            {
            LogMessage(LOG_ERR, "FreeArea must be between 0 and 255");
            exit(0);
            }

       if ((TicketGlobalData(EnableSidTicket) & -0x1) != 0)
            {
            LogMessage(LOG_ERR, "EnableSidTicket must be 0 or 1");
            exit (0);
            }
```

```
    if ((TicketGlobalData(EnableTicket) & -0x1) !- 0);
        {
        LogMessage(LOG_ERR, "EnableTicket must be 0 or 1");
        exit(0);
        }

    if ((TicketGlobalData(EnableLocalAuth) & -0x1) != 0)
        {
        LogMessage(LOG_ERR, "EnablLocalAuth must be 0 or 1");
        exit (0);
        }
    }
/*
 *----------------------------------------------------------------------
 *
 * TICKET_DebugHooks
 *
 *     Check for debug hooks and execute if found.
 *
 * Results:
 *     None.
 *
 * Side Effects:
 *     None.
 *
 *----------------------------------------------------------------------
 */
 tatic void TICKET_DebugHooks(ClientData clientData, char *suffix,
                              HTTP_Request *reqPtr)
     {
     if(strcmp(suffix, "/ticketstatus") == 0)
       {
       DumpStatus(reqPtr);
       HT_FinishRequest(reqPtr);
       return;
       }
   HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
   HT_FinishRequest(reqPtr);
   return;
   }
/*
 *----------------------------------------------------------------------
 *
 * DumpStatus --
 *
 *     Dump the server's ticket stat's
 *
```

```
 * Results:
 *     None.
 *
 * Side effects:
 *     None.
 *
 *----------------------------------------------------------------------
 */
#define BUFSIZE 1024
static void DumpStatus(HTTP_Request *reqPtr)
{
    HTTP_Server *serverPtr = reqPtr->serverPtr;
    char tmp[BUFSIZE], timeStr[BUFSIZE];
    struct utsname sysinfo;
    time_t uptime;
    int hours;

    HTTP_BeginHeader(reqPtr, "200 OK");
    HTTP_SendHeader(reqPtr, "Content-type: text/html", NULL);
    HTTP_EndHeader(reqPtr);
    HTTP_Send(reqPtr, "<title>WebServer Ticket Status</title>",
                        "<h1>WebServer Ticket Status</h1>:, NULL);

    HTTP_Send(reqPtr, "<p><hr>><p><h2>Ticket Log</h2>", "<p><pre>\n", NULL);

    sprintf(tmp, "    <b>%s:  </b>  %d\n", "Number of access          ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "    <b>%s:  </b>  %d\n", "Number of SID URL's       ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Valid SID's     ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Expired SID's   ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Invalid SID's   ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of XDomain accesses ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Local Redirects  ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Remote Redirects ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of No Auth servers  ", Ticket

    HTTP_Send(reqPtr, tmp, "</pre>", NULL);

    uptime = time (NULL) = serverPtr->started;
    uname(&sysinfo);
```

```
    striftime(timeStr, BUFSIZE, "%A, %d-%b-%y %T",
        localtime(serverPtr->started));

    springf(tmp, "Server runing on <d>%s</b> (%s %s) port %d, has been up \
                    since %s.<p>", sysinfo.nodename, sysinfo.sysname,
                      sysinfo.release, serverPtr->server_port, timeStr);
    HTTP_Send(reqPtr, tmp, NULL);:

    sprintf(tmp,."     <b>Number of connections:        </b> %d\n",
              serverPtr->numConnects);
    HTTP_Send(reqPtr, tmp, "<p><pre>\n", tmp, NULL);
    sprintf(tmp, "     <b>Number of HTTP requests:      </b> %d\n",
    HTTP_Send(reqPtr, tmp, "</pre><p>", NULL);

    hours = max(uptime / 3600, 1);
    sprintf(tmp, "This server is averaging <b>%d</b> requests per hour.<p>",
              serverPtr->numRequests/hours);
    HTTP_Send(reqPtr, tmp. NULL);

    DumpRusage(reqPtr);
/*     DumpConnections(reqPtr); */

    DNS_DumpStats(reqPtr);

    HTTP_Send(reqPtr, "<p><hr><address>", DStringValue(&ht_serverSoftware),
              "</address>\n", NULL);

    reqPtr->done = TRUE;

}
#undef BUFSIZE
```

【図１】

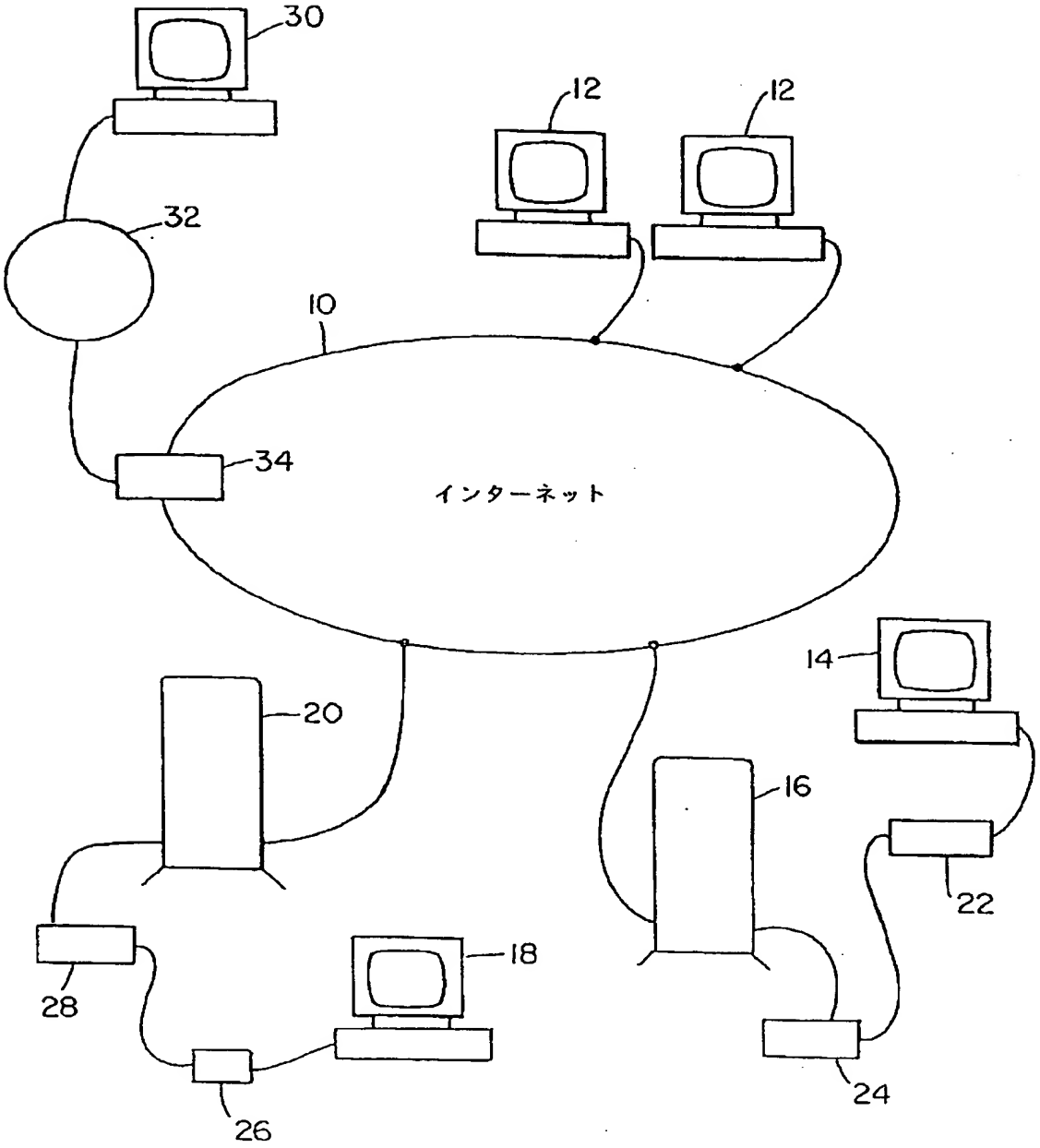

FIG. 1

【図２】

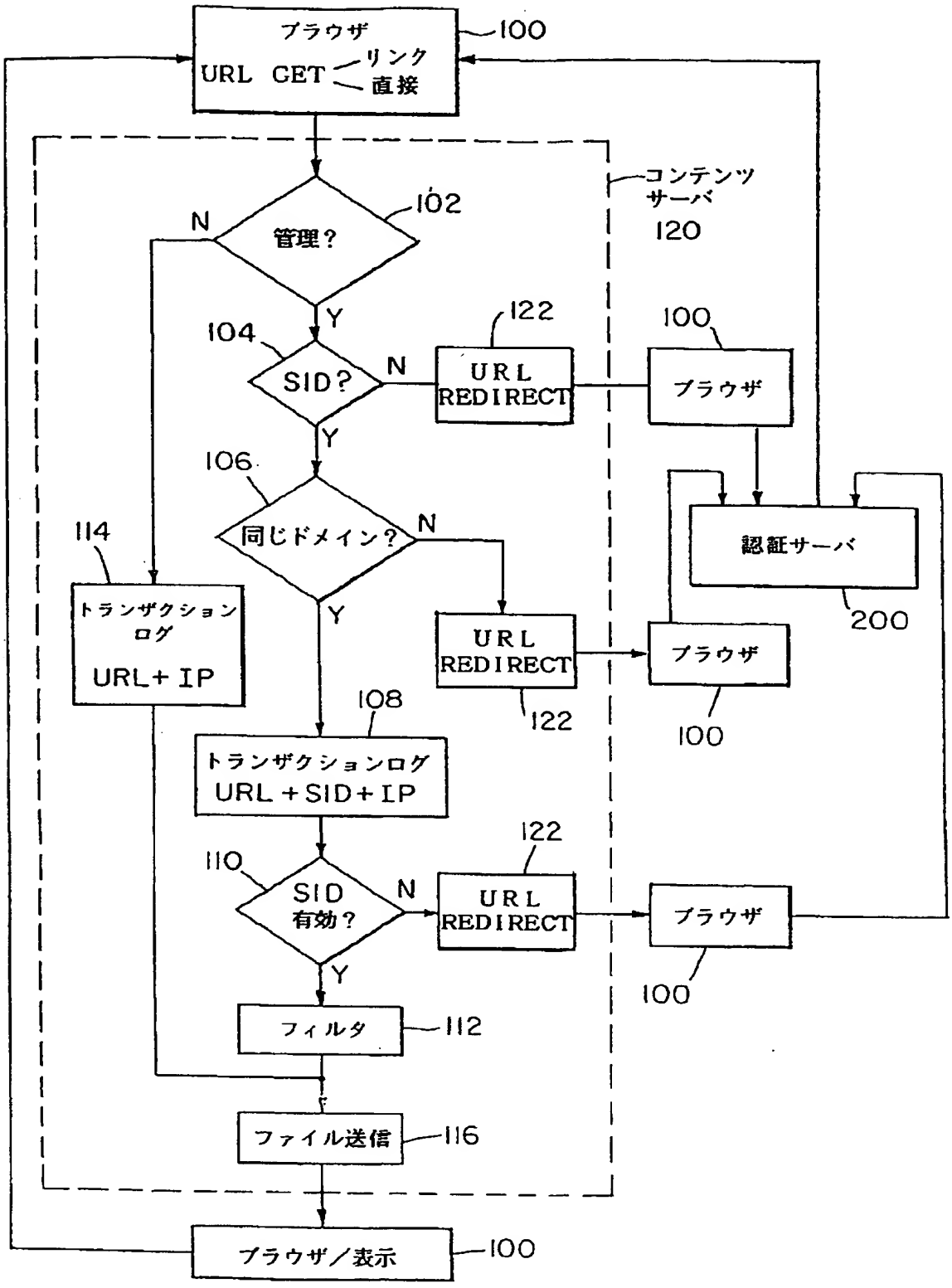
ブラウザ
URL GET
リンク
直接
100
コンテンツ
サーバ
120
102
管理？
N
Y
104
SID？
N
Y
122
URL
REDIRECT
100
ブラウザ
106
同じドメイン？
N
Y
114
トランザクション
ログ
URL＋IP
認証サーバ
200
URL
REDIRECT
122
ブラウザ
100
108
トランザクションログ
URL＋SID＋IP
110
SID
有効？
N
Y
122
URL
REDIRECT
ブラウザ
100
フィルタ
112
ファイル送信
116
ブラウザ／表示
100

FIG. 2A

【図２】

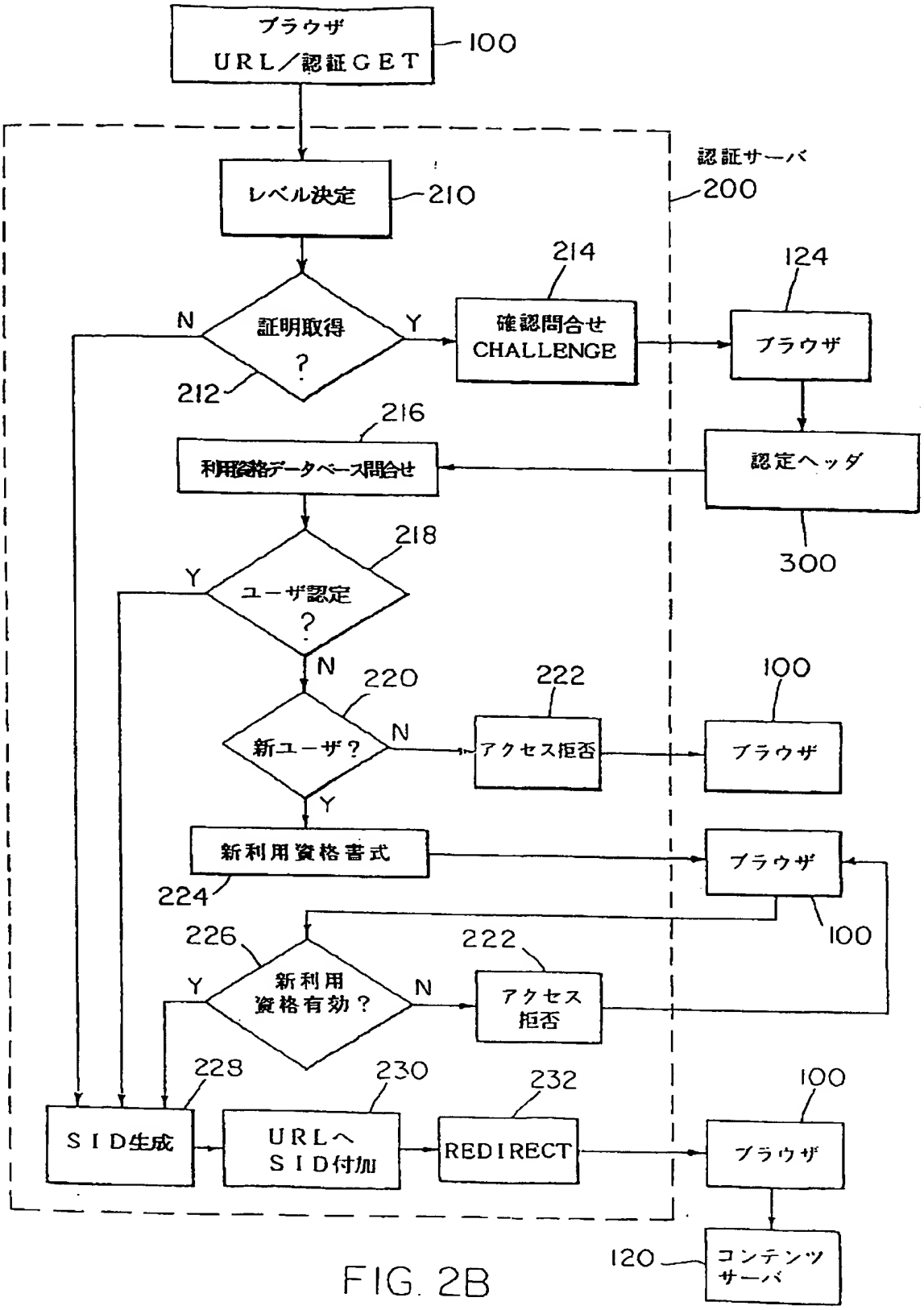
ブラウザ
URL／認証ＧＥＴ
100
認証サーバ
200
レベル決定
210
214
124
N
証明取得
？
Y
確認問合せ
CHALLENGE
ブラウザ
212
216
利用資格データベース問合せ
認定ヘッダ
300
218
Y
ユーザ認定
？
N
220
222
100
新ユーザ？
N
アクセス拒否
ブラウザ
Y
新利用資格書式
ブラウザ
224
100
226
222
Y
新利用
資格有効？
N
アクセス
拒否
228
230
232
100
ＳＩＤ生成
URLへ
ＳＩＤ付加
REDIRECT
ブラウザ
120
コンテンツ
サーバ

FIG. 2B

【図３】

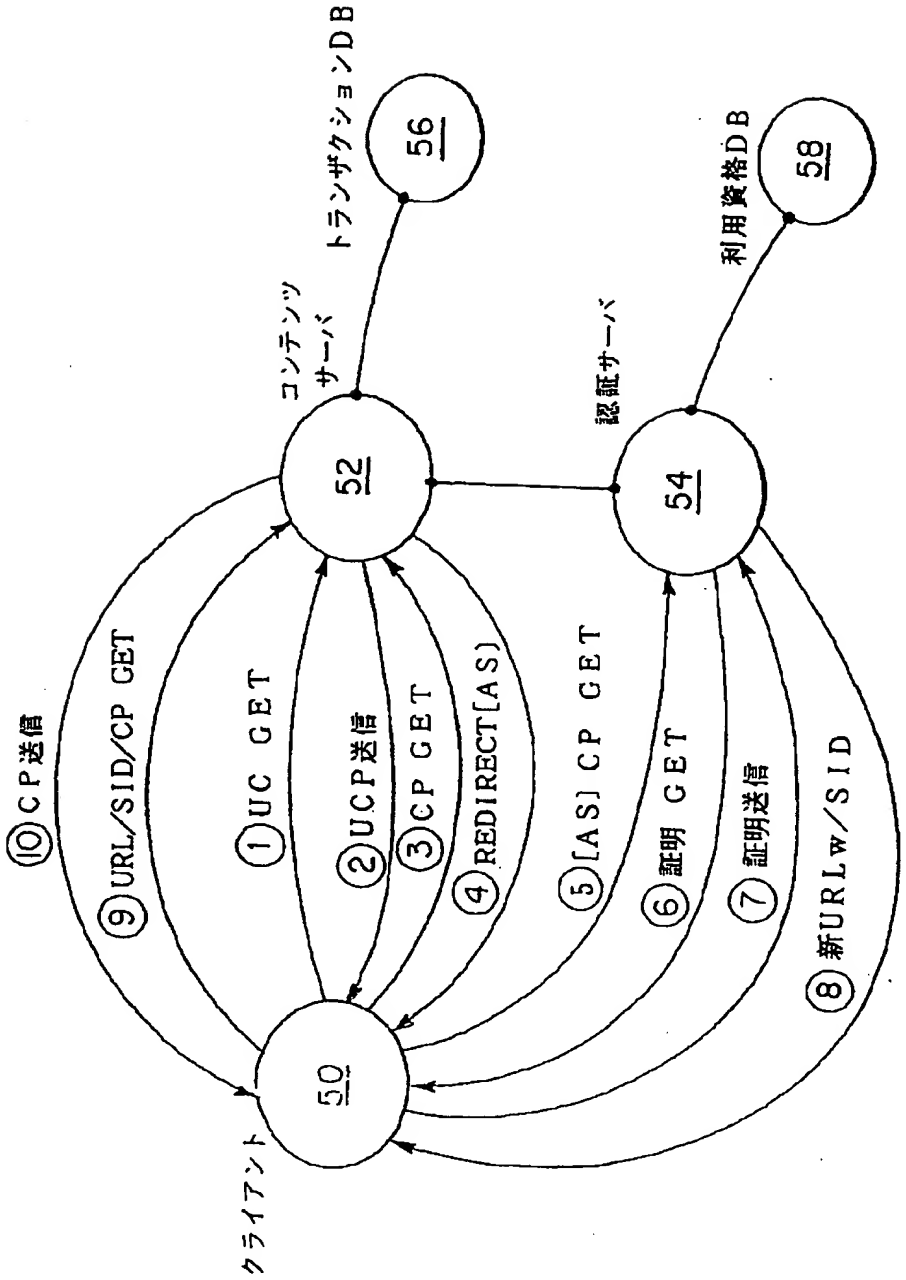
トランザクションDB
56
利用資格DB
58
コンテンツサーバ
52
認証サーバ
54
50
クライアント
①UC GET
②UCP送信
③CP GET
④REDIRECT[AS]
⑤[AS] CP GET
⑥証明 GET
⑦証明送信
⑧新URLw/SID
⑨URL/SID/CP GET
⑩CP送信

FIG. 3

【図４】

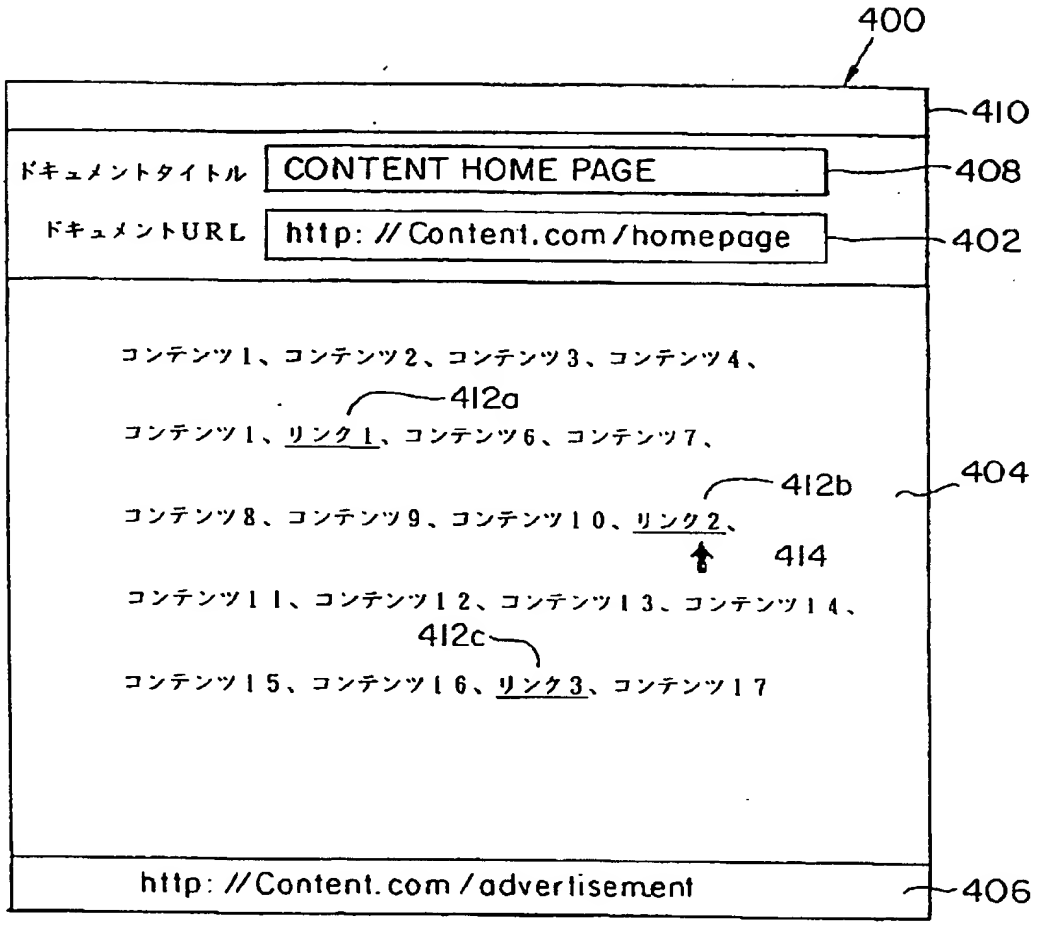
400
410
ドキュメントタイトル
CONTENT HOME PAGE
408
ドキュメントURL
http://Content.com/homepage
402
コンテンツ１、コンテンツ２、コンテンツ３、コンテンツ４、
412a
コンテンツ１、リンク１、コンテンツ６、コンテンツ７、
412b
404
コンテンツ８、コンテンツ９、コンテンツ１０、リンク２、
414
コンテンツ１１、コンテンツ１２、コンテンツ１３、コンテンツ１４、
412c
コンテンツ１５、コンテンツ１６、リンク３、コンテンツ１７
http://Content.com/advertisement
406

FIG. 4

【図５】

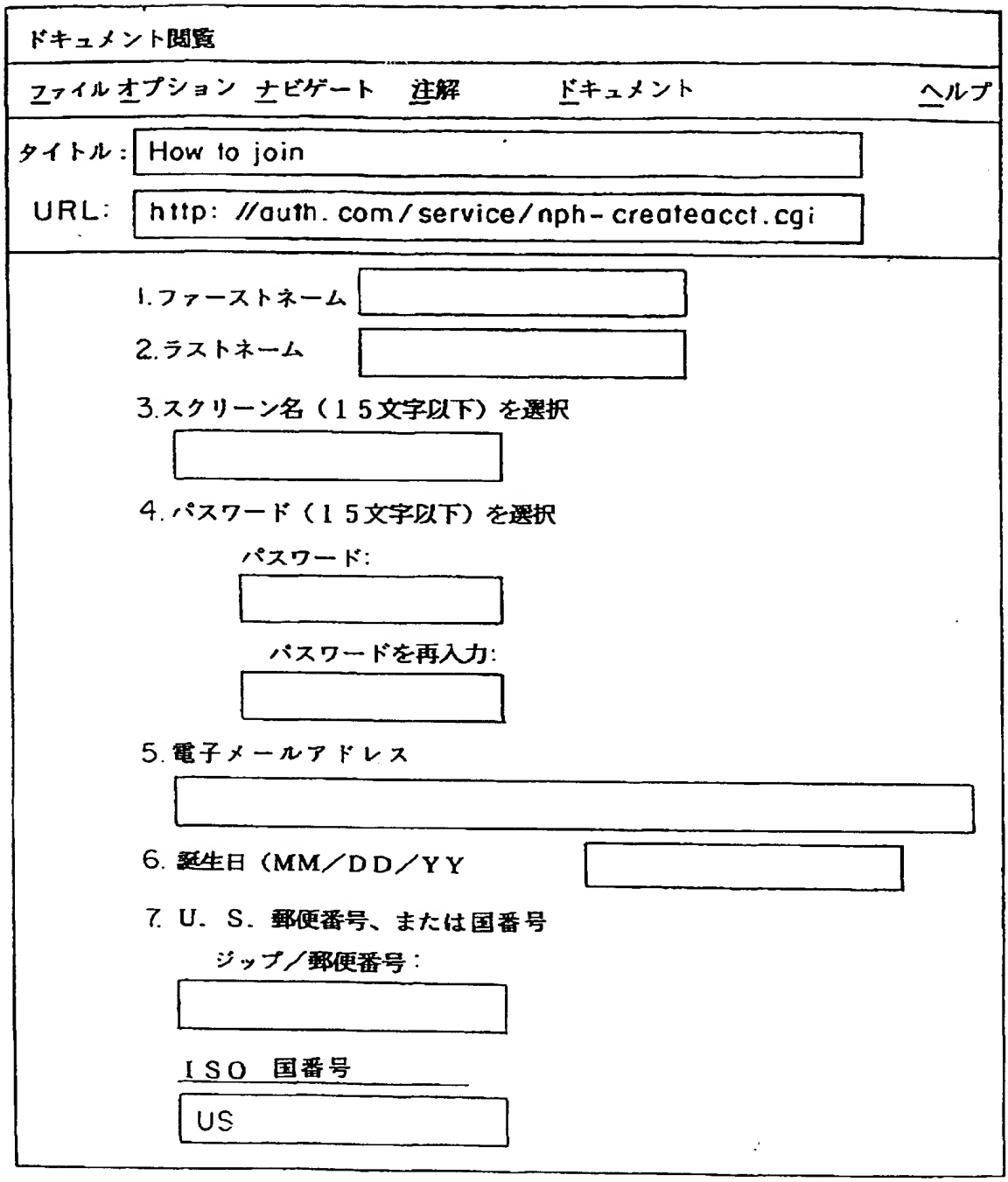
ドキュメント閲覧
ファイル オプション ナビゲート 注解 ドキュメント ヘルプ
タイトル: How to join
URL: http://auth.com/service/nph-createacct.cgi
1.ファーストネーム
2.ラストネーム
3.スクリーン名（１５文字以下）を選択
4.パスワード（１５文字以下）を選択
パスワード:
パスワードを再入力:
5.電子メールアドレス
6.誕生日（MM／DD／YY
7.U.S.郵便番号、または国番号
ジップ／郵便番号:
ISO 国番号
US

FIG. 5

【図６】

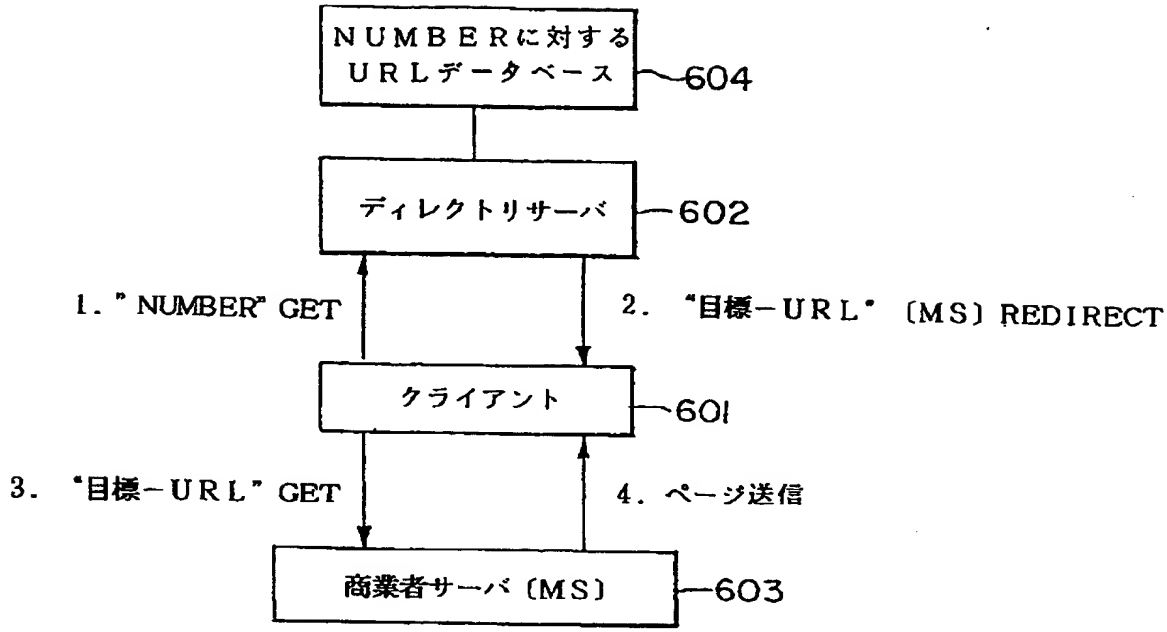
ＮＵＭＢＥＲに対する
ＵＲＬデータベース
604
ディレクトリサーバ
602
1．" NUMBER" GET
2．"目標－ＵＲＬ" 〔MS〕 REDIRECT
クライアント
601
3．"目標－ＵＲＬ" GET
4．ページ送信
商業者サーバ〔MS〕
603

FIG. 6

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 96/07838

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 456 920 (IBM) 21 November 1991 see page 5, line 33 - page 7, line 13	1,10
Y A		2,4-8 3,9, 11-44

Y	IEEE MULTIMEDIA, vol. 1, no. 1, 1994, COMPUTER SOCIETY US, pages 37-46, XP000440887 see page 43, left-hand column, line 12 - right-hand column, line 25	2,5

Y	EP,A,0 645 688 (KPN NEDERLAND) 29 March 1995 see column 1, line 37 - column 4, line 20 see figure 1	6-8

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 January 1997	04.02.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Canosa Areste, C

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 96/07838

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	IEEE NETWORK: THE MAGAZINE OF COMPUTER COMMUNICATIONS, vol. 9, no. 3, May 1995, NEW YORK US, pages 12-20, XP000505280 A.K.CHOUDHURY ET AL: "COPYRIGHT PROTECTION FOR ELECTRONIC PUBLISHING OVER COMPUTER NETWORKS" see page 14, right-hand column, line 49 - page 16, left-hand column, line 18 ---	4
A	WO,A,94 03959 (INTERNATIONAL STANDARD ELECTRIC CORP.) 17 February 1994 see page 5, line 25 - page 9, line 12 -----	1-44

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter nal Application No
PCT/US 96/07838

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-456920	21-11-91	US-A-	5560008	24-09-96
		JP-A-	3009444	17-01-91
		JP-B-	7060426	28-06-95
EP-A-645688	29-03-95	NL-A-	9301633	18-04-95
WO-A-9403959	17-02-94	DE-A-	4239754	03-02-94
		AU-A-	4703193	03-03-94
		DE-U-	9216139	18-02-93

フロントページの続き

(51)Int.Cl.[6]	識別記号	FI	
H04L	12/54		
	12/58		
	29/08		
H04L		9/00	673A
G06F		15/40	310C

(81)指定国　EP(AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), AU, CA, DE, GB, IL, JP

(72)発明者　スチュワート・ローレンス・シー
アメリカ合衆国, マサチューセッツ州 01803, バーリントン, アーバーウッド ドライブ　1

(72)発明者　モリス・ステフェン・ジェフリー
アメリカ合衆国, マサチューセッツ州 01886, ウエストフォード, キングス　パイン　ロード　3

(72)発明者　ペイン・アンドリュー・シー
アメリカ合衆国, マサチューセッツ州 01773, リンカーン, ルイス　ストリート　5

(72)発明者　ツリーズ・ジョージ・ウィンフィールド
アメリカ合衆国, マサチューセッツ州 02164, ニュートン, サコ　ストリート 81

(72)発明者　ギフフォード・デイビッド・ケー
アメリカ合衆国, マサチューセッツ州 02193, ウェストン, ピジョン　ヒル　ロード　26